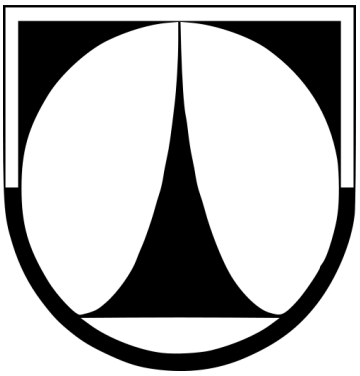
Technická univerzita v Liberci

Fakulta mechatroniky, informatiky a mezioborových studií



DIPLOMOVÁ PRÁCE

Uživatelské rozhraní pro přístup k LDAP

User's Interface for Access to LDAP

Autor práce: Bc. Jan Strnad

Vedoucí práce: Mgr. David Kmoch

LIBEREC

2009

Originál zadání

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé diplomové práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum: 25. 05. 2009

Podpis:

Poděkování

Tímto bych chtěl poděkovat Mgr. Davidu Kmochovi za pomoc, hodnotné rady a odborné vedení během mé práce.

Abstrakt

Tato diplomová práce řeší návrh a realizaci terminálového rozhraní k adresářové struktuře LDAP. Jádrem práce je implementace vyhledávacích a editačních funkcí, které umožní přístup k objektovým položkám LDAP serveru. Pro zobrazení výsledků těchto funkcí do dvoupanelové struktury je použito pseudografické rozhraní TUI (text user interface). Práce je rozdělena do dvou částí. První vysvětluje základní pojmy a principy přístupu ke struktuře LDAP, analyzuje problém a simuluje podmínky nasazení na Technické univerzitě v Liberci. Zakončena je výběrem vhodných prostředků pro implementaci. Druhá část pak popisuje návrh modelu aplikace a implementační detaily.

Klíčová slova: LDAP, ncurses, TUI, ASN.1, Base64

Abstract

The diploma work deals with the proposition and realization of console interface to the LDAP structure. The core of the work is the implementation of retrieval and editing functions that enable the access to object items of the LDAP server. The pseudo-graphic interface TUI (text user interface) is used for the display of the results of the functions to the dual-panel structure. The diploma work is divided into two parts. The first part explains the basic terms and principles of the access to the LDAP structure**,** it analyzes the issue and simulates the conditions of implementation to the Technical University of Liberec. At the end of the first part of the work, we can find the choice of convenient instruments for the implementation. The second part describes the proposition of the model application and the implementation details.

Keywords: LDAP, ncurses, TUI, ASN.1, Base64

Obsah

1 Úvod

Tato diplomová práce se zabývá návrhem a implementací pseudografického klientského programu pro přístup k LDAP. Konkrétně jde o tvorbu aplikace, která usnadní administrátorovi přístup k adresářové struktuře. Cílem této práce je seznámit se s adresářovou strukturou LDAP a podání přehledu o současném stavu problematiky. Dále pak tvorba klientského konzolového programu s TUI a podrobný popis použitých metod a kroků.

Téma této diplomové práce vychází ze zadání semestrálního projektu „Terminálové rozhraní pro vyhledávání v LDAPu“, který byl zaměřen pouze na zpracování výsledků vyhledávání (tedy nadstavba vyhledávací funkce ldapsearch). Aplikace byla programována v jazyce C a jako základ pseudografického rozhraní byla použita knihovna ncurses. Při vypracování projektu se úspěšně simuloval běh aplikace na nezabezpečeném LDAP serveru, na zabezpečeném školním serveru však aplikace nefungovala. Dále aplikace obsahovala některé těžko odhalitelné chyby přístupu k paměti při použití pokročilejších částí knihovny ncurses. Tyto nedostatky budou v diplomové práci vyřešeny a zároveň dojde k implementaci vyhledávacích a editačních funkcí. Při programování aplikace, na rozdíl od projektu, bude využit objektově orientovaný jazyk C++, vybrané části knihovny STL a jako pseudografické rozhraní se použijí některé open source rozšíření knihovny ncurses, případně bude naprogramována knihovna vlastní pro zpřehlednění práce s programem pomocí dialogových oken.

Zpracované téma je přehledně rozděleno do úvodu, tří kapitol a závěru. První kapitola je seznámením se základy dané problematiky – tedy pojmy jako například LDAP, DN, DIT, OID, filter atd. Rovněž je tato kapitola věnována kódování ASN.1 a normám RFC, které nám umožní lépe se orientovat v dané problematice. Klíčová část této kapitoly je věnována analýze dostupných prostředků. Jsou zde porovnány přístupy k adresářové službě, klientské i serverové aplikace.

Druhá kapitola se zabývá návrhem aplikace. Jsou zde uspořádány základní požadavky na daný program a jeho funkce. Dále je navržen model programu a schémata základních částí.

Poslední kapitola je věnována praktické části této práce. Je zde popsána simulace lokálního serveru LDAP, podrobný popis programovaných částí a v závěru jsou okomentovány snímky obrazovky z reálného provozu.

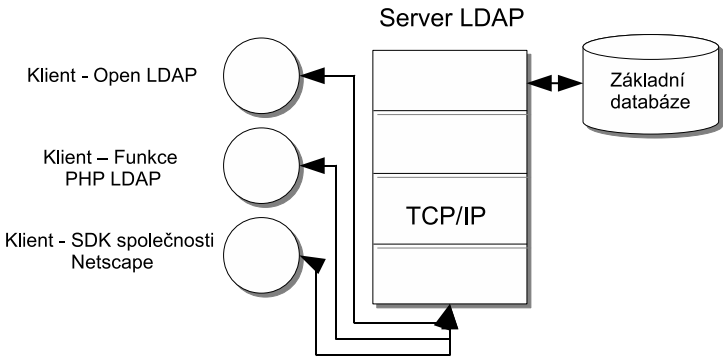
2 Současný stav řešené problematiky

První část této kapitoly se zabývá úvodem do problematiky adresářové struktury, jejími základními pojmy s přehlednými schématy, obecnému pojednání o struktuře LDAP a historickým přehledem. V další části je popisován princip kódování Base64, základy formálního jazyka ASN.1 a přehled norem vztahujících se k dané problematice. Poslední část se věnuje funkcionálnímu modelu LDAP, metodám přístupu a základnímu přehledu klientských aplikací.

2.1 Vysvětlení základních pojmů

Obecně o LDAP

LDAP (Lightweight Directory Access Protocol) – je velmi komplexní síťová služba pro k přístup adresářové struktuře. Zdroj [7] definuje LDAP jako *„protokol pro ukládání a přístup k datům na adresářovém serveru. Podle tohoto protokolu jsou jednotlivé položky na serveru ukládány formou záznamů a uspořádány do stromové struktury (jako ve skutečné adresářové architektuře). Je vhodný pro udržování adresářů a práci s informacemi o uživatelích (např. pro vyhledávání adres konkrétních uživatelů v příslušných adresářích, resp. databázích).“* Protokol LDAP vznikl odvozením od mezinárodního telekomunikačního standardu ICCTT[1] X.500. Tento protokol (DAP) však používal objemný síťový model ISO/OSI a mnoho nadbytečných funkcí spojených s transakcemi – stal se tedy velmi obtížně implementovatelný. Na základě praktických zkušeností byl původní DAP, jak je z názvu patrné, „odlehčen“. LDAP implementuje jednoduchý a velmi rozšířený protokol TCP/IP.



Obrázek 2.1: Komunikace Klient/Server

[1] International Consultative Commitee of Telephony and Telegraphy

Dle [6] LDAP z protokolu X.500 převzal databázový a bezpečnostní model a tento protokol dále rozšířil, aby bylo možné využívat existující internetové bezpečnostní standardy jako SASL a SSL/T viz níže. Na ilustraci překreslené z [6] je vidět, jak klienti prostřednictvím protokolu TCP/IP a serveru LDAP komunikují s databází. Původní verze LDAPu byly jen prostředníky mezi klienty a protokolem DAP. Dnes již pojem LDAP zahrnuje i samotnou adresářovou strukturu, ale pro klienta zůstává přístup stále stejný.

V současné době existuje několik velmi úspěšných implementací tohoto protokolu. Asi nejznámější z nich je open source projekt OpenLdap dostupný na `http://www.openldap.org`. K dispozici je pod licencí GPL a poskytuje funkcionalitu jak serverové, tak klientské části. Z tohoto důvodu bude využit pro testovací účely. Podrobnější přehled implementací serveru a klienta je uveden v části 2.5. Velmi přesný popis LDAPu můžeme získat nahlédnutím do RFC 4511 [12]. RFC sice není obecný standard (je to vlastně jen doporučení), zato je však dodržován u všech implementací vyjma implementace Microsoft Active Directory. Ta se od RFC 4511 odchyluje. Více o RFC naleznete v kapitole 2.3.

Historie LDAP

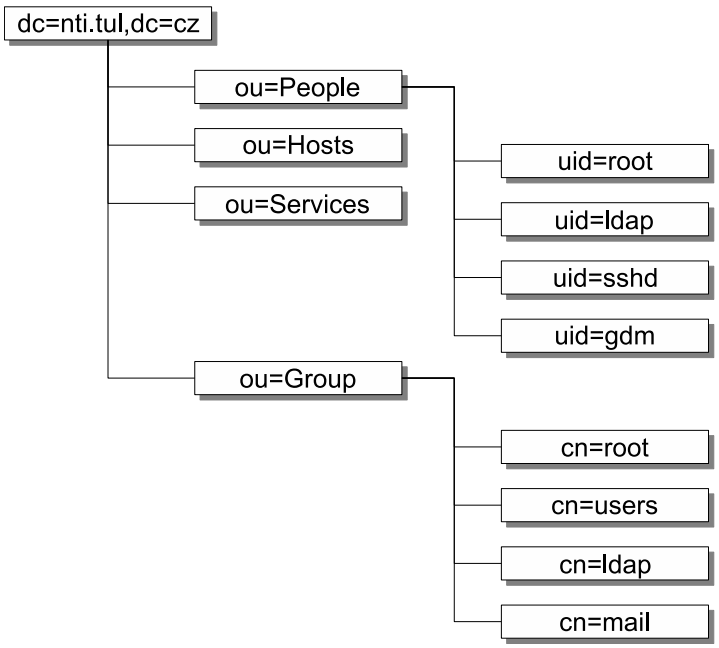
Tento protokol byl vytvořen ve spojených státech na Michiganské univerzitě. V roce 1993 byla uveřejněno RFC 1487 s popisem LDAP Verze 1. Následně vyšly RFC 1488 s popisem reprezentace řetězců a atributů a RFC 1558 s popisem vyhledávacího filtru. Přehled verzí lze dohledat v RFC 4511 (nebo jeho v předchůdci RFC 2251). Jsou to následující verze:

- Verze 1 (1993) - zastaralá a dnes již nepoužívaná verze, byla navržena jen jako prostředník mezi klientem a protokolem X.500.
- Verze 2 (1995) - je definována RFC 1779, většina dnešních LDAP serverů i klientů je kompatibilní s touto verzí, ale implicitně bývá tato kompatibilita zakázána[2]. Oproti novější verzi je méně restriktivní – některé z definic jsou uvozeny slovem „může“ (may) místo „musí“ (must). Výsledná implementace je závislá na programátorovi.
- Verze 3 (1997) - je v dnešní době používána ve všech aktuálních implementacích LDAP. Podle [6] tato verze podporuje nový bezpečnostní model, k zajištění zabezpečených transakcí používá ověřování pomocí vrstvy SASL (Simple Autentification an Security Layer – jednoduchá ověřovací a zabezpečovací vrstva). Ta dovoluje velkou pružnost při výběru správného

[2] V případě použití OpenLdap se dá povolit v konfiguračním souboru slapd.conf (popis v části 4.1) vložením hodnoty allow bind_v2.

ověřovacího schématu. Velmi často se k tomuto účelu používá protokol SSL (Secure Socket Layer – vrstva zabezpečených soketů) a jeho následovník TLS (Transport Layer Security – zabezpečení provozní vrstvy).

DIT (Directory Information Tree) – tedy organizace do stromové struktury. Podle [1] se pod DIT *„rozumí především konkrétní návrh struktury adresářového stromu, jeho větvení, tj. rozčlenění položek a informací jimi nesených do hierarchicky uspořádaných skupin.“* Podobně jako v citovaném zdroji je níže překreslena část DIT s referenčními daty. Bude využita pro popis základních částí stromu:



Obrázek 2.2: DIT referenčních dat

Hlavou tohoto stromu je kořenový prvek zvaný *root suffix*, v tomto případě tedy *dc=nti.tul,dc=cz*. Asi jako v každém informačním stromu jsou jednotlivé entity rozlišovány jako uzly (node) a listy (leaf). Platí, že nositelem informace je jak list, tak uzel. Všechny prvky tohoto stromu jsou jednoznačně identifikovatelné, a to i v globálním měřítku. Platí zde analogie s doménovým názvem na internetu. LDAP využívá stejného principu, jen místo tečkové notace používá čárku. Tento identifikátor se pak nazývá rozlišovací jméno.

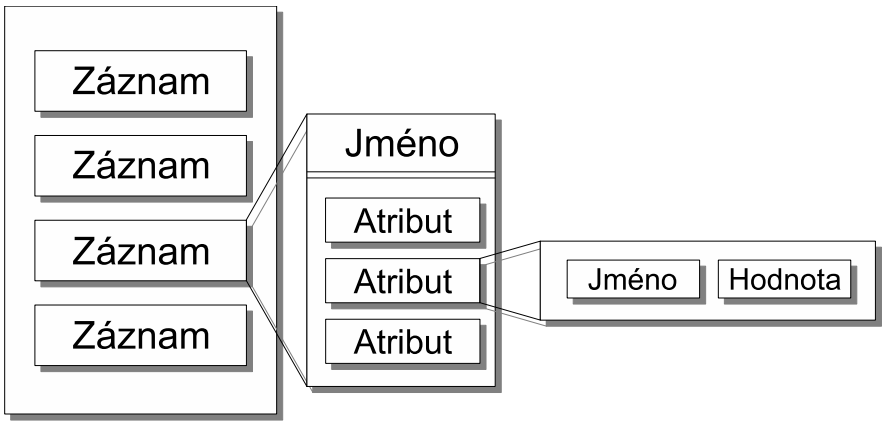
Rozlišovací jméno (Distinguish Name) – často zkracováno na DN. V rámci jedné větve (branch) lze dále definovat tzv. **relativní rozlišovací jméno** (Relative Distinguish Name), jenž je unikátní jen v rámci jedné části stromu adresářové struktury. Rozlišovací jméno je pak spojením jednotlivých relativních rozlišovacích jmen, které tvoří cestu ke kořenovému prvku. Na referenčních datech si vybereme list *cn=root* větve *ou=Group*.

V rámci všech potomků uzlu *ou=Group* je RDN *cn=root* unikátní. Pokud budeme chtít získat DN tohoto listu, musíme postupovat metodou „zdola nahoru“, tedy přidáme za RDN listu RDN předka a oddělíme čárkou. Výsledné DN je tedy *cn=root,ou=Group,dc=nti.tul,dc=cz*.

Báze (Base) – je vstupní bod pro prohledávání. Jedná se o rozlišovací jméno a je vždy potřeba při komunikaci s vyhledávacím modulem i při samotném úvodním spojení se serverem. S vyhledáváním souvisí též pojem **rozsah prohledávání** (Search scope) určující, které záznamy budou odpovídat vyhledávání a **Filtr** (Filter) jako prostředek pro specifikaci výběrového kritéria. Tyto pojmy budou podrobněji rozebrány v kapitole 2.4.

Záznam (Entry) – někdy se tento pojem zaměňuje za objekt, kontejner či svazek. Záznam je základní entita adresářové struktury. Celý strom (DIT) je tvořen záznamy a každý záznam má rozlišovací jméno a obsahuje soubor atributů. Objekt, stejně jako objektová třída, je základní součástí záznamu – je jimi tvořen.

Atributy (Atribute, attr) – jednotlivé atributy jsou vlastnostmi záznamu, tj. jsou nositeli dat. V množině atributů jednoho záznamu má každý atribut jedinečný název. Mohou obsahovat informace jako jsou jméno, telefonní číslo, email, cestu k domovskému adresáři atd. Každý atribut (jeho datový typ) je součástí schématu 2.1, dále je zde definována citlivost na velikost písmen (case sensitivity) a počet výskytu atributu. Pokud je povolen vícenásobný výskyt, hovoříme o tzv. mutlihodnotě (multivalue). Multihodnotou jsou ve své podstatě atributy se stejným názvem, ale rozdílnou hodnotou, někdy se zapisují jako jeden atribut s více hodnotami oddělenými čárkou.



Obrázek 2.3: Vztah mezi záznamem, atributem a jeho hodnotou

Ze zdroje [1] bylo volně převzato a překresleno znázornění vztahu mezi záznamem a atributem. Obrázek 2.3 ukazuje seznam záznamů pořízených např. vyhledávací funkcí ldap_search. Každý záznam obsahuje své jméno (RDN) a neprázdnou množinu atributů. Obecně platí, že každý záznam musí obsahovat atribut *objectclass*.

Ve schématu je pak definováno, pro jakou hodnotu tohoto atributu platí jaká omezení – více viz *2.1.* Ze zdroje byla [3] volně přeložena a přizpůsobena tabulka běžně se vyskytujících atributů a tabulku pro určování relativního rozlišovacího jména.

Tabulka 2.1: Běžné atributy LDAP

Syntaxe	Popis
bin	binární informace
ces	řetězec citlivý na velikost znaků, známý jako adresářový řetězec
cis	řetězec bez citlivosti na velikost znaků
tel	telefonní číslo, čísla jsou brány jako text.

Klíčový rozdíl mezi syntaxí cis a ces se projeví během porovnávání řetězců. U cis se na velikost písmen nebere zřetel. U syntaxe tel je zajímavé, že je nastaven vstupní filtr. Všechny „bílé znaky“[3] a pomlčky jsou vynechány. Tabulka níže dává přehled o základních atributech pro určování relativního rozlišovacího jména.

Tabulka 2.2: Příklady atributů pro RDN

Jméno atributu (alias)	Syntaxe	Popis	Hodnota
domain component (dc)	cis	část doménového jména	cz
organization (o)	cis	jméno organizace	tul
organizationalUnit (ou)	cis	jméno organizační jednotky	People
commonName (cn)	cis	běžné označení	root
jpegPhoto	bin	obrázek ve formátu JPEG	fotografie uživatele Jiří_Novák

Na obrázku 2.2 je patrné, že každý prvek tohoto stromu je pojmenován právě pomocí těchto základních atributů. Ty pak tvoří relativní rozlišovací jméno prvku. Například kořenový prvek má dva atributy *dc=nti.tul* a *dc=cz,* které tvoří jeho RDN.

Schéma – na schéma lze nahlížet jako na soubor pravidel. Schéma definuje **objektovou třídu** (object class), tj. pravidla pro výskyt povolených záznamů a atributů.

[3] angl. blank, definováno v IEEE Std 1003.1-2001.

Podle zdroje [1] *„třída objektu definuje jména a typy atributů, které se mohou v objektu (položce) vyskytovat, přičemž určuje, které z nich jsou povinné (musí se vyskytovat). Typ položky ve formě jména třídy objektu je u každé položky uveden ve vyhrazeném a povinném atributu objectClass.“* Někdy je též pojem schéma brán jako návrh informačního modelu (tj. výběr vhodných tříd objektů, návrh DIT).

Sufix – je část rozlišovacího jména, která je společná všem záznamům v DIT.

LDIF (LDAP Data Interchange Format) – neboli formát výměny dat LDAP je lidmi čitelný formát dat LDAP. Právě komunikace mezi klientem a serverem probíhá přes protokol LDAP (tedy binární protokol využívající zápis ASN.1 a jeho binární podobu **BER**). Podle zdroje [6] byl jazyk LDIF definován s ohledem na „lehkost“ LDAP. LDIF je textový formát, přičemž i binární data převádí metodou **Base64** (popisovanou níže **2.2**) do textové podoby a ukládá jako součást definice LDIF. Základní forma LDIFu je uvozena DN záznamem ve tvaru: *dn: <rozlišovací jméno>* následována seznamem atributů ve tvaru: *<typ atributu>: <hodnota atributu>*. Příklad tohoto zápisu byl vytvořen exportem záznamu *uid=root,ou=People,dc=nti.tul,dc=cz* a *uid=sshd,ou=People,dc=nti.tul,dc=cz* do formátu LDIF:

```
dn: uid=root,ou=People,dc=nti.tul,dc=cz
uid: root
cn: root
sn: root
objectClass: person
...
homeDirectory: /root

dn: uid=sshd,ou=People,dc=nti.tul,dc=cz
uid: sshd
cn: sshd
sn: sshd
objectClass: person
...
homeDirectory: /var/empty/sshd
```

Pokud hodnota atributu začíná mezerou nebo dvojtečkou, případně obsahuje jiný znak než US-ASCII, je převedena do notace Base64. Více záznamů v jednom LDIF je odděleno novým řádkem.

LDIF je jako formát spojený s protokolem LDAP standardizován pomocí RFC 2849. Některé aplikace však využívají původní formu tzv. UMich verze, která nepodporuje komentáře (značeno symbolem „#“).

2.2 Kódování

Jádrem této diplomové práce je tvorba klientské aplikace přistupující k objektům LDAP. A právě z této praktické části (Kapitola 4) vyplynul požadavek na teoretickou znalost kódování Base64 a formálního jazyka ASN.1.

Base64

Base64 je kódování obecných binárních dat. Provádí se za pomoci tabulky (převzata z [2]) Base64, která mapuje 64 čísel na znaky a dodatečný symbol =, značící výplň konce textu (zarovnání). Kóduje se každá šestice bitů, ne běžných 8, proto vzrůstá počet výsledných znaků přibližně o třetinu.

Pro zakódování 64 znaků je potřeba minimálně 6 bitů (ze vztahu $2^6 = 64$). Vstup je nejprve rozdělen na segmenty po 24 bitech a ty na 4 šestice. Dále jsou pomocí tabulky překódovány a doplněny o znak = (maximálně však 2). Výstup musí být uspořádán do řádek dlouhých maximálně 76 znaků. Využití tohoto kódování je opravdu široké. Zdroj [8] uvádí, že bylo poprvé definováno normou RFC 989 v roce 1987 jako podpora pro PEM (Privacy-Enhanced Electronic Mail). Pro nás bude jistě důležité, že vyhledávací funkce LDAPu (ldapsearch) vrací své výsledky ve formátu LDIF. Ten může obsahovat pouze znaky ze sady ASCII. V případě, že atribut obsahuje i jiné znaky, je jeho hodnota vypsána v kódování Base64 RFC 1521 odst. 5.2.

Tabulka 2.3: Kódovací tabulka

Hodnota	Kód	Hodnota	Kód	Hodnota	Kód	Hodnota	Kód
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v	výplň	=
14	O	31	f	48	w		
15	P	32	g	49	x		
16	Q	33	h	50	y		

Posledním znakem v tabulce je speciální ukončovací znak =. Na malém příkladě si teď ukážeme, jak kódovaný text vypadá. S mezivýsledky budeme kódovat „Ahoj Base64“.

Tabulka 2.4: Postupné kódování

Vstup	Kódováno base64
„Ahoj“	QWhvag==
„Ahoj “	QWhvaiA=
„Ahoj b“	QWhvaiBi
„Ahoj ba“	QWhvaiBiYQ==
...	...
„Ahoj Base64“	QWhvaiBCYXNlNjQ=

Pro tento příklad bylo využito kódovacích schopností OpenSSL [10] a text upraven pomocí příkazu:

```
openssl enc -base64 -in vstupni_soubor -out vystupni_soubor
```

Princip kódování nejlépe ukáže tabulka převodu slova Ahoj:

Tabulka 2.5: Detailní přehled

Vstup 8-bit	01000001	01101111	01101010

Vstup 6-bit	010000	010110	100001	101111	011010	100000		
Výplň							00	00
Dekadicky	16	22	33	47	26	32	=	=
Výstup (Base64)	Q	W	h	v	a	g	=	=

Všimněte si, že vstupní text byl rozdělen po šesti bitech, ale poslední šestice byla neúplná, a proto byla doplněna dvěma binárními nulami. Ve výsledku jsou přidány dva znaky =.

ASN.1

Pro lepší pochopení následujících kapitol je nutné, abychom se seznámili s formálním jazykem ASN.1. Setkat se s tímto jazykem je velmi snadné - je používán ve velkém množství průmyslových a mezinárodních norem z oblasti bezpečnosti. Tento formální jazyk (uznaný i jako česká norma) dokáže popsat libovolné abstraktní datové struktury. Jedna z nejdůležitějších vlastností je nezávislost na procesoru i operačním systému - říkáme, že jazyk je tzv. multiplatformní. Díky této vlastnosti umožňuje správnou interpretaci dat a jejich výměnu mezi počítači, celými platformami nebo vrstvami ISO/OSI.

ASN.1 (Abstract Syntax Notation One) - zdroj [9] jej uvádí jako jazyk umožnující definování abstraktních objektů. V následující části uvedu podle RFC 4511 [12] a RFC4512 definici nového typu DistinguishedName:

```
1 DistinguishedName ::= RDNSequence --alias
```

DistinguishedName je tedy jen jiné označení pro RDNSequence.

```
2 RDNSequence ::= SEQUENCE OF RelativeDistinguishedName
```

Ve shodě s úvodní definicí rozlišovacího jména je tedy DistinguishedName (DN) tvořeno sekvencí RelativeDistinguishedName (RDN).

```
3 RelativeDistinguishedName ::= SET SIZE (1..MAX) OF
4       AttributeTypeAndValue
```

RDN pak obsahuje jeden až MAX konstruovaného typu AttributeTypeAndValue (atribut), přičemž MAX je celočíselná hodnota definovaná též v RFC 4511.

```
5 AttributeTypeAndValue ::= SEQUENCE {
6      type  AttributeType,
7      value AttributeValue }
```

A nyní definice základních typů:

```
8  AttributeType ::= LDAPString
9  AttributeDescription ::= LDAPString
10 AttributeValue ::= OCTET STRING
11 LDAPString ::= OCTET STRING
```

Na příkladu si můžeme všimnout typického zápisu ASN.1, tedy sled datových typů a hodnot. Na první pohled je syntaxe velmi podobná programovacímu jazyku. Za zmínku též stojí, že se nejprve uvádí proměnná (resp. její identifikátor), a poté typ.

Třídy typů – základním kamenem tohoto jazyku jsou datové typy. V české normě ČSN ISO/IEC 8824 ASN.1 rozeznáváme čtyři třídy:

1. Universal - pro typy, které jsou definovány přímo normou ASN.1 Tyto typy mohou být používány kdekoliv (ve všech normách a aplikacích), někdy se jim též říká typy vestavěné.

2. Application - pro datové typy, jejichž význam je specifický pro určitou aplikaci, například typy z normy X.500.

3. Private - pro typy, jejichž význam je specifický v rámci nějaké úzce specializované struktury (firemní, školní atd.).

4. Context-specific - pro typy, jejichž význam je specifický pro daný konstruovaný (složený) typ. Slouží k odlišení komponent strukturovaných typů.

Konvence – typy třídy UNIVERSAL se píší velkými písmeny, identifikátory konstruovaných typů se píší s velkým písmenem na začátku. Komentář (např. použitý v definici DistinguishedName) se odděluje dvěma pomlčkami --. Pomocí jednoduchých typů lze definovat typy složitější. Vnitřně je každý typ tvořen dvojicí třída a hexadecimální číslice. Každá základní třída má svůj rozsah přidělených hodnot. Univerzální typy mají rozsah 0x01 až 0x1E. Například typ BOOLEAN je vnitřně zapsán jako UNIVERSAL 1.

V příloze 6 se uvádí úplný výpis vestavěných typů mezinárodní normy X.680 [11]. Názvy jednoduchých typů jsou velmi přehledné, jak uvádí doktor Klíma v publikaci [9] *„Například BIT STRING je libovolný řetězec bitů, IA5 STRING je libovolný řetězec znaků mezinárodní abecedy číslo 5 (ASCII), INTEGER je libovolné celé číslo, NULL je prázdná hodnota, OCTET STRING je libovolný řetězec oktetů (osmic bitů)“*

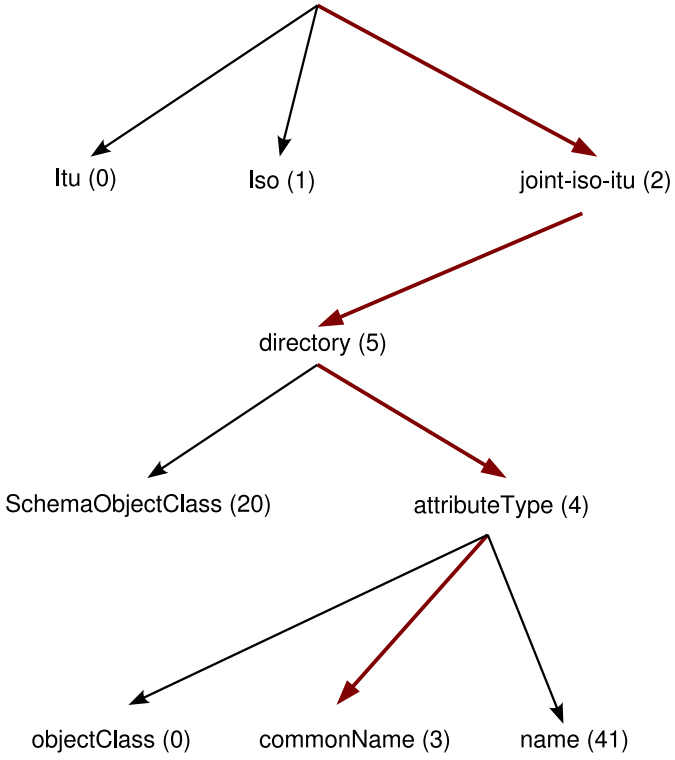
Specifikace dovoluje používání zkratek pro jednotlivé typy. Toto je důležité u typů, které mají více slov. V praxi se tedy zapisují jako BMPString, IA5String, UTCTime a podobně.

Objektové identifikátory – jsou zvláštním prostředkem jazyka ASN.1, vnitřně jsou definovány jako UNIVERSAL 6 a využívají se jako odkaz do jiné normy nebo souboru definic. Struktura odkazu je sled čísel (jednotlivých komponent). Dovolím si teď ukázat výstup z programu *dumpasn1* při zpracování (dekódování jazyka ASN.1) binárních dat komunikace protokolu LDAP.

```
1  209   28:    SET {
2  211   26:      SEQUENCE {
3  213    9:        OBJECT IDENTIFIER commonName (2 5 4 3)
4  224   13:        LDAPString 'root'
5          :        }
6          :      }
```

Použitím Objektového identifikátoru (*OBJECT IDENTIFIER*) se v tomto případě z obyčejného typu *LDAPString* stává *commonName* neboli cn. Zajišťuje to sled komponent {2 5 4 3}.

Objektové identifikátory jsou vydávány registrovanými organizacemi a tvoří digitální strom. Pokud chceme získat identifikátor registrované položky, stačí nám uvést hodnoty hran směrem od kořene. Zde je uveden konkrétní, ale neúplný strom atributu cn. Slovní vyjádření se používá pouze pro přehled, důležitá jsou čísla hran zpravidla uváděná v kulatých závorkách za názvem. Celému sledu se potom říká OID.



Obrázek 2.4: Strom objektů

Konstruované typy – jsou významově podobné jako například struktury v jazyce C - slouží k zapouzdření základních nebo rozšiřujících typů. Konstruují se pomocí operátoru ::= a typu. V ASN.1 existují čtyři konstruované typy:

1. SEQUENCE - uspořádaný seznam jednoho nebo více typů
2. SEQUENCE OF - uspořádaný seznam žádného nebo více výskytů jednoho daného typu
3. SET - uspořádaný seznam jednoho nebo více typů
4. SET OF - uspořádaný seznam žádného nebo více výskytů jednoho daného typu

Jednotlivé položky se potom oddělují čárkou tak, jak je vidět z příkladu definice konstruovaného typu AttributeTypeAndValue a jeho položek AttributeType a AttributeValue.

Ostatní typy – ASN.1 rozeznává ještě jeden druh typů, a to typy variantní, někdy též nazývané zástupné.

1. CHOICE - přebírá typ z vybrané položky. Je definován pomoci klíčového slova CHOICE a výčtu variant.

2. ANY - je libovolný typ známý až v době použití, tento přístup dovoluje například implementovat algoritmy, které ještě nejsou známé.

Kódování ASN.1 – ASN.1 lze kódovat více způsoby, mezi základní možnosti patří BER, CER, DER, XER a PER. Kódování je v tomto případě prostředek, který z textové podoby ASN.1 udělá podobu binární. Asi nejčastěji používané kódování je **BER** (*Basic Encoding Rules*). Pro tento způsob kódování je typické, že lze shodnou proměnnou kódovat více způsoby. Příkladem může být rozdílné kódování obyčejného typu OCTET STRING a konstruovaného typu skládajícího se pouze z OCTET STRING. Dalším typem kódování je **DER** (*Distinguished Encoding Rules*). Jedná se o zúžení pravidel BER a jak je už z názvu patrné, kódování je jednoznačné. Tedy jedna hodnota může být kódována jen jedním způsobem. Pro ASN.1 ještě existují další typy kódování, velmi rozšířené je jiné zúžení BER, a to **CER** (Canonical Encoding Rules), nebo **XER** (XML Encoding Rules) - zápis ASN.1 pomocí značkovacího jazyka XML.

2.3 Normy

V předcházejícím textu se vyskytují časté odkazy na velké množství norem, standardů a doporučení. Cílem této podkapitoly je uvést seznam těch nejdůležitějších v dané problematice a krátce popsat jejich obsah.

RFC

Na otázku, co to RFC vlastně je, není jednoduché odpovědět. Parafráze zdroje [13] RFC definuje jako anglickou zkratku *„request for comments*“, tedy „žádost o komentáře“. Jak již název napovídá, RFC jsou oficiálně považovány spíše za doporučení než normy v tradičním smyslu, přesto se podle nich řídí drtivá většina Internetu. RFC se tedy používá jako označení řady standardů a dalších dokumentů popisujících internetové protokoly, systémy apod.

Všechna RFC jsou volně ke stažení na adrese `http://www.ietf.org/rfc.html`, případně na dalších zrcadlech[4], kde bývají dostupné ve formátu ASCII, PDF a HTML.

Zajímavý je podle [13] i vznik normy: Původními autory jednotlivých RFC jsou obvykle konkrétní experti, kteří se snaží řešit konkrétní problém – řešení nabídnou ve formě návrhu RFC internetové veřejnosti (jako tzv. Internet Draft). Pokud je dané řešení (často již dobře fungující v rámci nějakého pilotního provozu) uznáno za přínosné, dokument se vydá jako RFC.

Pro naši potřebu zde uvedu pouze ty, které se úzce dotýkají oblasti LDAPu a přidružených definic. Při podrobném studiu zjistíte, že v každé z uvedených norem jsou důležité odkazy na další normy (RFC), a proto by byl jejich úplný seznam velmi dlouhý.

- RFC 4510 Lightweight Directory Access Protocol (LDAP) Technical Specification Roadmap – je velmi užitečný rozcestník s odkazy na aktuální normy. Nahradil předchozí rozcestník RFC 3377. RFC 4510 je dostupný na <`http://tools.ietf.org/html/rfc4510`>

- RFC 4511 Lightweight Directory Access Protocol (LDAP): The Protocol – popisuje základní části protokolu LDAP, jejich význam i zápis (kódování). Definuje obálku zprávy (message envelop) – každá operace s protokolem LDAP je zapouzdřena touto obálkou. Dále definuje základní řetězcové typy, atributy, autentizační, aktualizační i dotazovací operace. Poslední část je věnována bezpečnostnímu modelu. Nahrazuje RFC 2251, 2830, 3771. RFC 4511 je dostupný na <`http://tools.ietf.org/html/rfc4511`>

- RFC 4512 Lightweight Directory Access Protocol (LDAP): Directory Information Models – se zabývá adresářovým informačním modelem. Popisuje syntaxi základních částí LDAPu jako je schéma, objectClass aj. v definičním jazyku ABNF (Augmented Backus-Naur Form). RFC 4512 je dostupný na <`http://tools.ietf.org/html/rfc4512`>

- RFC 4513 Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms – tedy autentifikační metody a zabezpečovací mechanismy. Tento RFC definuje připojování k LDAP a zabezpečení metodou TLS. RFC 4513 je dostupný na <`http://tools.ietf.org/html/rfc4513`>

[4] angl. mirror, znamená odlišné umístění se stejným obsahem

- RFC 4514 Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names – popisuje řetězcovou reprezentaci rozlišitelného jména za pomoci formálního jazyka ASN.1. Též řeší převody a zobrazení znaků nepatřících do sady US-ASCII (např. diakritika). RFC 4514 je dostupný na <`http://tools.ietf.org/html/rfc4514`>
- RFC 4515 Lightweight Directory Access Protocol (LDAP): String Representation of Search Filters – řetězcová reprezentace vyhledávacího filtru. Obdobně jako RFC 4514 definuje pomocí formálního jazyka ASN.1 strukturu vyhledávacího filtru a možnosti jeho zápisu. RFC 4515 je dostupný na <`http://tools.ietf.org/html/rfc4515`>
- RFC 4516 Lightweight Directory Access Protocol (LDAP): Uniform Resource Locator – neboli LDAP URL. Tato norma specifikuje formát URL pro LDAP v.3. RFC 4516 je dostupný na <`http://tools.ietf.org/html/rfc4516`>
- RFC 4517 Lightweight Directory Access Protocol (LDAP): Syntaxes and Matching Rules – se zabývá formátem hodnot atributů, tj. definuje pro každý základní typ masky ve formátu ABNF. RFC 4517 je dostupný na <`http://tools.ietf.org/html/rfc4517`>
- RFC 4519 Lightweight Directory Access Protocol (LDAP): Schema for User Applications – je integrální částí technické specifikace protokolu LDAP. Pomocí ABNF zavádí atributy konstruované z atributů základních a přiřazuje jim OID. RFC 4519 je dostupný na <`http://tools.ietf.org/html/rfc4519`>

2.4 Analýza dostupných prostředků

V této podkapitole se zaměřím na rozbor prostředků přístupu k objektovým položkám LDAPu.

Poskytovaná funkcionalita

Správný postup před návrhem aplikace, jenž využívá externí funkcionality, vyžaduje podrobnou analýzu. Každá z implementací LDAPu se podle doporučení RFC 4511 dělí na autentizační operace a aktualizační a dotazovací operace. Představíme si oba druhy operací a popíšeme:

- Autentizační operace
  - bind – je operace sloužící pro připojení uživatele do adresářové služby.

  - unbind – tato operace není opačná k bind, jak by se na první pohled zdálo[5], slouží k ukončení operace.
  - abandon – okamžité přerušení prováděných operací na serveru, tj. zrušení nedokončené operace.

- Aktualizační a dotazovací operace
  - search – tento modul prohledává strukturu záznamů a vrací jeden nebo více záznamů.
  - add – přidávání záznamů do struktury.
  - delete – mazání záznamů.
  - compare – slouží pro porovnávání záznamů.
  - modify – úprava záznamů ve struktuře.

Autentizační operace – Tyto operace slouží k navazování případně rušení spojení se serverem LDAP. Při navazování spojení je podle RFC 4511 možno zvolit způsob přihlašování, a to třemi způsoby:

- **Anonymous bind** – tedy anonymní přihlášení, kdy je odesláno prázdné DN i heslo. Toto přihlášení pak dovolí uživateli přistoupit k v veřejným položkám serveru LDAP.

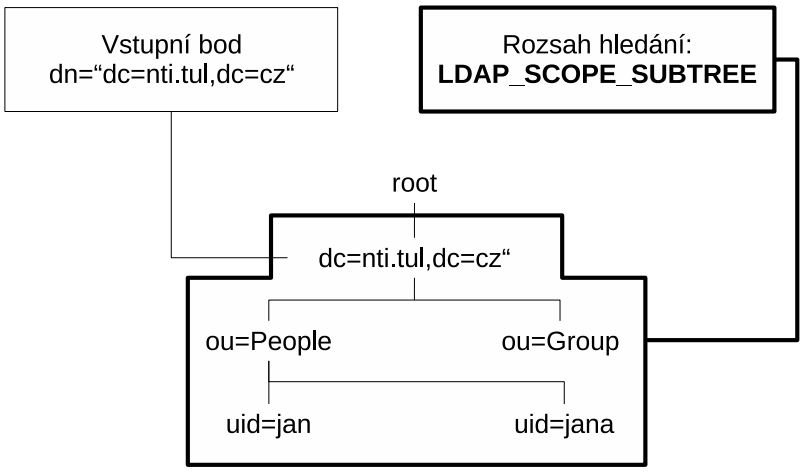
- **Simple bind** – je přihlášení pomocí rozlišovacího jména uživatele a hesla. Heslo je přenášeno v nešifrované podobě, tzv. plaintext.

- **SASL bind** – Simple Authentication and Security Layer neboli obecná metoda přidávání nebo zlepšování zabezpečení. Pomocí této metody lze provést autentifikaci klienta na server LDAP různými bezpečnostními mechanismy (například mechanismem Kerberos nebo jako v případě serveru ldap.tul.cz pomocí certifikátů a mechanismu TLS).

Dotazovací operace – V RFC 4511 je definována jako obecná struktura dotazování klienta na server, hotové aplikace a externí knihovny. Ty pak zpravidla obsahují implementace LDAPu. Konkrétně u OpenLdapu se implementace dotazovací operace jmenuje ldapsearch, případě ldapsearch.exe (API funkce ldap_search). Tato utilita sestaví tzv. Search Request neboli vyhledávací požadavek (opět je zde použit jazyk ASN.1) a odešle jej serveru. Ten, pokud úspěšně proběhlo ověření identity, vrátí výsledek ve formě záznamů (formát LDIF). V následující části pomocí zdroje [3] a [5] projdeme možnosti ldapsearch a nalezneme důležité parametry, které pomohou při návrhu výsledné aplikace. Vyhledávací požadavek má následující vstupní parametry:

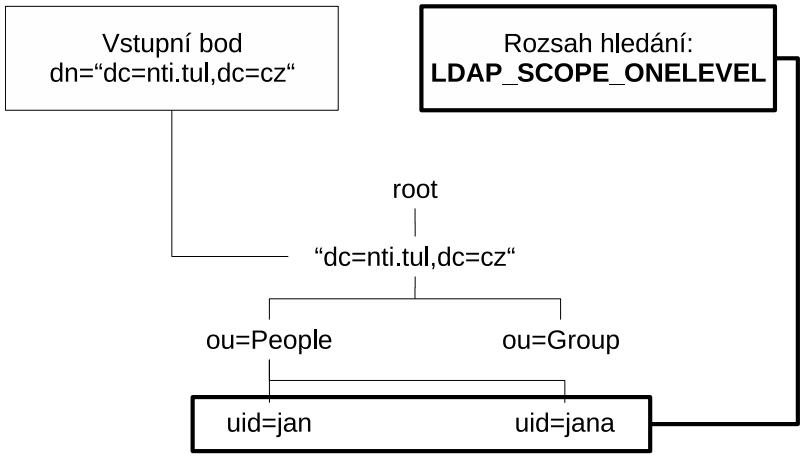
[5]Podle RFC 4511 pochází název unbind z historie LDAP.

- **baseObject** – tedy výše definovaný pojem báze. Umožňuje zúžit prohledávání na určitou pod-větev, a to zadáním jejího rozlišovacího jména (DN).
- **scope** – určuje hloubku prohledávání. K dispozici jsou tři možnosti (grafy jsou volnou úpravou zdroje[5]):
  - **sub** (LDAP_SCOPE_SUBTREE) – pro implementaci OpenLdap je tento přepínač implicitně definován hodnotou 1. Při použití tohoto přepínače (nebo pokud přepínač není uveden vůbec) vrací vyhledávací modul ldapsearch kompletní stromovou strukturu, která vyhovuje zadanému filtru. Struktura je procházena rekurzivně a výsledky řazeny sestupně podle umístění v adresářové struktuře.



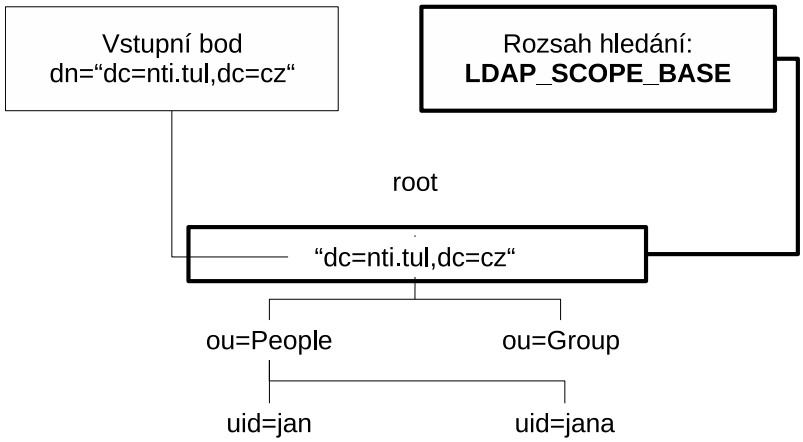
Obrázek 2.5: Subtree

  - **onelevel** (LDAP_SCOPE_ONELEVEL) – tento přepínač je obdoba přepínače LDAP_SCOPE_SUBTREE s tím rozdílem, že jsou vráceny výsledky pouze poslední úrovně vyhovující zadaným hodnotám vstupního bodu a filtru. Tento stav se hodí v případě, že je výsledků prohledávání mnoho. Již na straně serveru se výsledky omezí na jednu úroveň, a výrazně tím přispěje ke zvýšení rychlosti komunikace se serverem.



Obrázek 2.6: Onelevel

– **base** (LDAP_SCOPE_BASE) – je stav přepínače scope, který vrátí výsledky omezující se pouze na zadaný vstupní bod (base). Tento přepínač je prospěšný při získávání iniciálních dat o stromu záznamů, tj. zobrazení kořene vyhledávání a jeho atributy.



Obrázek 2.7: Base

- **derefAliases** – řeší přístup k aliasům a jejich případný překlad. Povolené jsou následující hodnoty: neverDerefAliases, derefInSearching, derefFindingBaseObj, derefAlways.
- **sizeLimit** – udává maximální počet výsledků vyhledávací funkce, podle [6] je pro OpenLdap výchozí hodnota tohoto parametru 100 a při dosažení limitu vrací chybu LDAP SIZELIMIT EXCEEDED.
- **timeLimit** – časové omezení pro přijímání výsledků vyhledávací funkce. Nastavuje se v sekundách a při dosažení limitu vrací chybu LDAP TIMELIMIT EXCEEDED.

- **typesOnly** – je booleová hodnota. Pokud je nastavena na true vrací ldapsearch pouze informace o typech.
- **filter** – jeho syntaxe je následující: (atribut operátor hodnota). Definici filtrů a jejich návrhem se zabývá RFC 2254[4] které filtr definuje pomocí jazyka ASN.1 Předpokládejme strom, jenž ve své větvi obsahuje záznam „uid“ a ten obsahuje podzáznamy a, ..., Jan, Jana, ..., z. Jako povolený operátor můžeme uvést:

Tabulka 2.6: Použití filtru

Název	Operátor	Příklad	Výsledek
Přesné znění (equalityMatch)	=	(uid=Jan)	uid=Jan
Větší nebo rovno (greaterOrEqual)	>=	(uid>=Jan)	uid=Jan,uid=Jana až uid=z
Menší nebo rovno (lessOrEqual)	<=	(uid<=Jan)	uid=a až uid=Jan
Přibližně (approxMatch)	~=	(uid~=Jan)	uid=Jana
Přítomnost (present)	=*	(uid=*)	uid=a až uid=z

Pomocí závorek lze vyhledávací podmínky logicky spojovat pomocí operátorů „&“ a „|“ (log. A a log. NEBO). Jako hodnotu lze uvést jakoukoliv definovanou objektovou třídu nebo znak „*“ znamenající, podobně jako např. u regulárních výrazů, všechny objekty. Zde jsou uvedeny příklady použití:

Tabulka 2.7: Filtr s log. operátory

Název	Operátor	Příklad	Výsledek
Logický součet (AND)	&	(& (uid=Jan) (email=*@tul.cz))	uid=Jan s atributem email končícím @tul.cz
Logický součin (OR)	|	(| (uid=Jan)(uid=Jana)	uid=Jan,uid=Jana
Logická negace (NOT)	!	(!(uid=Ja*))	Všechny záznamy které nezačínají Ja.

- **attributes** – je booleová hodnota. Pokud je nastavena na true, jsou výsledkem vyhledávání pouze jména atributů bez hodnot.

Aktualizační operace – Tyto operace poskytují možnost přistupovat k adresářové struktuře a měnit její obsah. Podle toho, k jakému druhu obsahu přistupují, se dají rozdělit na dvě kategorie:

- Práce se záznamem
  - Vytvoření záznamu
  - Změna relativního rozlišovacího jména
  - Změna umístění (odkaz na rodiče)
  - Smazání záznamu
- Práce s atributem
  - Přidání atributu určenému záznamu
  - Změna hodnoty atributu
  - Smazání atributu
  - Porovnání atributu

Při práci se záznamem je vždy zapotřebí rozlišitelné jméno případně jeho relativní část. U operací s atributy jsou vstupními parametry DN záznamu, který atribut obsahuje nebo obsahovat bude. Následuje jméno atributu a jeho hodnota. Ve vývojovém balíku OpenLdap existují API funkce modify, delete a add, přičemž funkce add se nahrazuje voláním funkci modify s jinými parametry. U operace porovnání jednotlivých atributů existuje funkce compare, která přijímá dva atributy a vrací hodnotu TRUE nebo FALSE podle nalezené shody.

2.5 Přehled softwarových produktů

První část této kapitoly se věnuje přehledu softwarových produktů s ohledem na implementovanou funkcionalitu a ovládání (zejména grafické). Běžný přístup k adresářové struktuře se dělí na 3 základní skupiny. Z každé této skupiny byl vybrán nějaký reprezentant:

Terminálové nástroje

Jako zástupce programů dostupných z příkazového řádku byla vybrána velmi rozšířená implementace OpenLdap API tedy programy:

- ldapadd – je odkaz na ldapmodify (hardlink)
- ldapcompare – porovnání atributů
- ldapdelete – mazání záznamů a atributů
- ldapmodify – úprava hodnot atributů
- ldapmodrdn – úprava relativních jmen záznamů

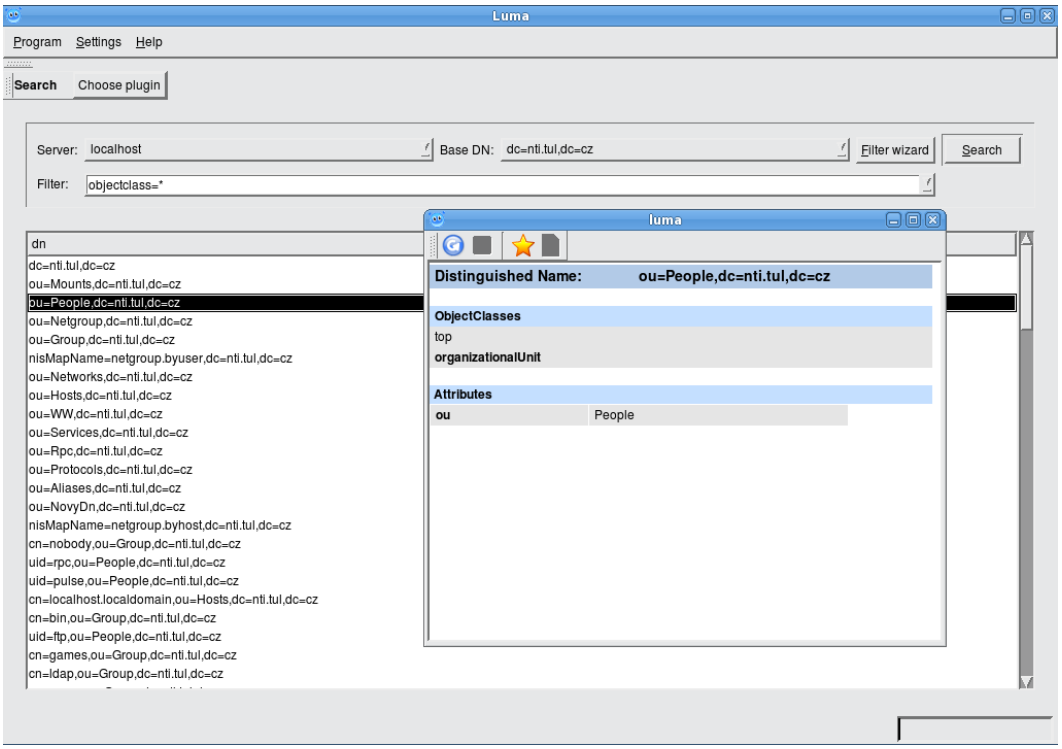
- ldapsearch – vyhledávání záznamů

Jejich nesporné výhody jsou rychlost přístupu a textový vstup i výstup na terminál, který může být dále zpracován, např. pomocí roury. Nevýhodou pak může být obtížná manipulace s velkým počtem výsledků vyhledávání a mnoho parametrů (přepínačů), které snižují uživatelskou přívětivost těchto nástrojů.

Grafické nástroje

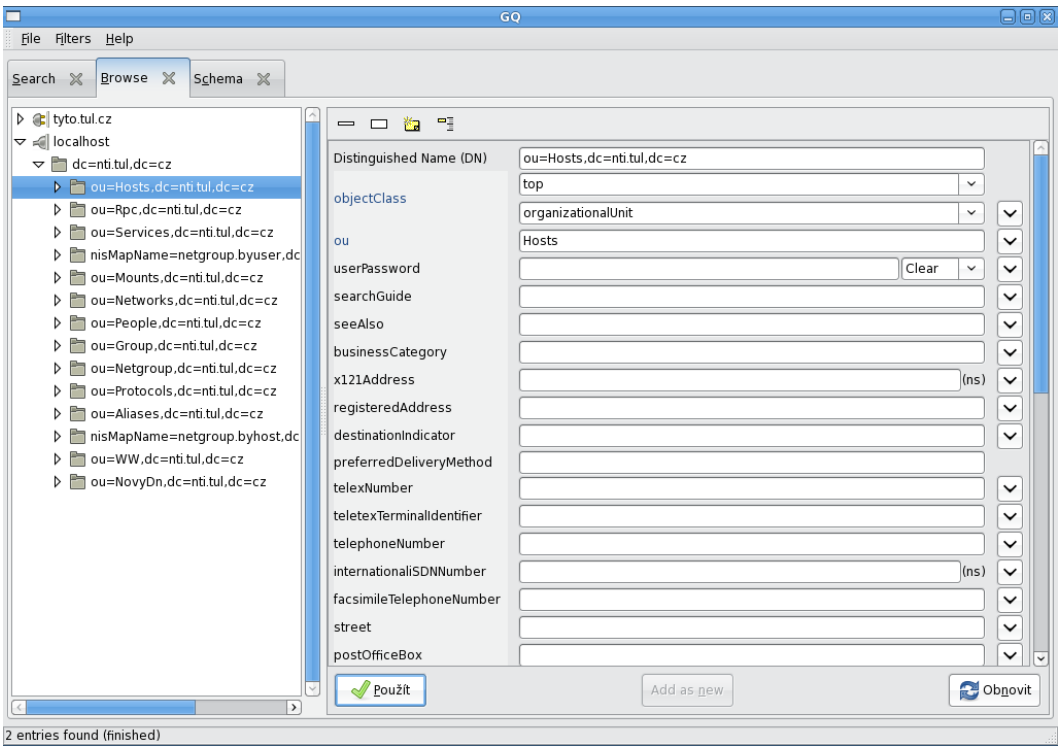
Tedy nástroje s grafickým uživatelským rozhraním. Reprezentanti této skupiny byli vybíráni pouze ze skupiny softwaru s otevřeným kódem, existují i uzavřené alternativy, ty jsou ovšem placené a jejich kód není k dispozici. Pro účel této diplomové práce byly u této kategorie zkoumány podporované funkce a rozložení grafických prvků na formulářích a dialogových oknech.

- **Luma** – Je projekt dostupný na adrese: `http://luma.sourceforge.net/`. Jedná se o klienta postaveného na grafickém toolkitu Qt4 firmy trolltech. Napsán je v jazyce Python a používá knihovnou PyQt.



Obrázek 2.8: Hlavní okno programu Luma a editační okno záznamu

- **Gq** – Tento klient je založen na grafickém toolkitu GTK+/GTK2 a programován v jazyce C++. Jeho vývoj se zastavil v roce 2006, a tak obsahuje mnoho známých neopravených chyb. Projekt je dostupný na adrese: `http://sourceforge.net/projects/gqclient`



Obrázek 2.9: Program Gq

- **Apache Directory Studio** – Odborníky velmi dobře hodnocený projekt. Tento software existuje jak ve verzi plug-inu vývojového prostředí eclipse, tak jako samotná distribuce. Projekt je dostupný na adrese: `http://directory.apache.org/studio/`

Webové aplikace – jsou často dodávány ke komerčním řešením LDAP

- Novell iManager – je dodáván k produktu Novell eDirectory Administration. Produkt je dostupný na adrese: `http://www.novell.com/products/edirectory/`

- phpLDAPadmin – implementace klienta LDAP v jazyku PHP. Projekt je dostupný na adrese:`http://phpldapadmin.sourceforge.net/wiki/index.php/Main_Page`

2.6 Vývojové nástroje

Zde je podán přehled základních možností programátora klientského softwaru pro přístup k LDAP:

- OpenLDAP - projekt OpenLDAP vychází z implementace University of Michigan. Při přístupu na stránky UMICH jste přesměrováni právě na stránky projektu OpenLDAP.

Jedná se o rychle se rozvíjející projekt poskytující server i vývojové nástroje volně šířené včetně zdrojových textů dostupný na adrese `http://www.openldap.org/`.

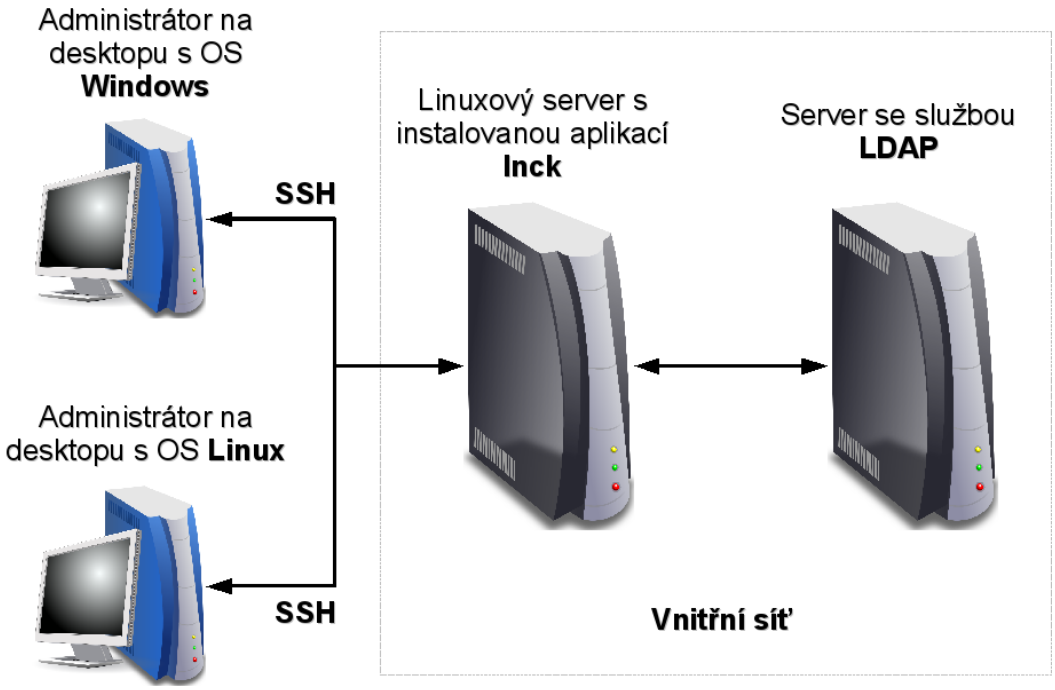
- Mozilla LDAP C SDK – velmi používaný soubor nástrojů a knihoven dostupný na adrese `http://www.mozilla.org/directory/csdk-docs`.
- Netscape Directory SDK for Java – je obdobou Mozilla LDAP C SDK pro programovací jazyk Java. `http://www.mozilla.org/directory/javasdk.html`.
- PerLDAP – modul pro jazyk Perl dostupný na adrese `http://www.perldap.org/`.
- Java Naming and Directory Interface (JNDI) - je produkt vyvíjený firmou Sun Microsystems. Jedná se o zobecněné rozhraní ke jmenným a adresářovým službám tedy přístup k DNS a LDAP. Narozdíl od Java SDK poskytuje abstrakci vyšší úrovně. Více viz `http://java.sun.com/products/jndi/`.

3 Návrh aplikace

Před samotným návrhem aplikace je třeba shrnout základní požadavky kladené na aplikaci a zhodnocení předcházející analýzy. Na jeho základě pak rozhodnout o použití konkrétního programovacího jazyka a externích knihoven. Dále navrhnout funkční model aplikace a sestavit jeho diagram.

3.1 Jak bude aplikace fungovat

Výsledná aplikace bude sloužit jako pomocný nástroj administrátora serveru LDAP. Pozorováním úkonů při administraci bylo odvozeno následující schéma:



Obrázek 3.1: Uživatel-klient-server

Podpůrná aplikace lnck[1] bude nainstalována (kompilována pro cílovou platformu) na jednom ze serverů vnitřní sítě. Administrátor se ze svého stolního počítače protuneluje na server s instalovaným programem lnck. Tedy použije zabezpečený komunikační protokol SSH.[2] Zde aplikaci spustí a může pohodlně přistupovat k LDAPu.

[1] Název výsledné aplikace je zkratkou Ldap New Curses Klient.
[2] Administrátor používající OS Windows může využít program PuTTY, administrátor s OS Linuxu příkaz ssh.

Od běžné práce výše zmíněného administrátora se liší pouze použitím programu lnck. Ten se vzdáleně přihlásil na linuxový server vnitřní sítě a zde využíval terminálové nástroje popisované v části 2.5. Výsledná aplikace má tyto nástroje nahradit a učinit práci s LDAP v terminálu více uživatelsky přívětivější.

3.2 Základní požadavky

Nejprve je nutné určit základní požadavky na vyvíjenou aplikaci. Ty mi později poslouží při návrhu implementace v oblastech, jako je výběr platformy, programovacího jazyku, formát konfiguračního souboru nebo použitý toolkit. Jako klíčové cíle, jenž má výsledná aplikace splňovat, byly konzultacemi stanoveny tyto body:

- **Přenositelnost** – v úvahu budou brány pouze programovací jazyky, které jsou multiplatformní. Pokud dále předpokládáme běh aplikace na operačním systému GNU/Linux, bylo by vhodné, aby byl výsledný produkt POSIX-konformní, což zajistí přenositelnost mezi různými verzemi operačních systémů Unixového typu.

- **Rychlost** – Program bude muset masivně přistupovat k přidělené paměti, a proto bude z hlediska rychlosti vykonávání programu vhodnější jazyk kompilovaný. Na jedné straně tento požadavek znevýhodní ve výběru jazyky interpretované. Na straně druhé bude největší zpoždění programu spočívat v komunikaci se serverem LDAP, a proto jsou jazyky interpretované a kompilované z hlediska rychlosti rovnocené. U výběru vhodného programovacího jazyka jsme tedy více než požadavkem na rychlost limitováni dostupnými externími knihovnami pro konkrétní jazyk.

- **Jednoduché a přehledné zobrazení výsledků vyhledávání** – nabízí se možnost využití nějaké externí knihovny pro zobrazeni v textovém režimu či nějaká vlastní implementace rozhraní. Další problematickou částí bude požadavek na možnost pohybu ve výsledcích, i ten vede na použití pseudografické nadstavby.

- **Implementace CLI** (Command Line Interface) – tedy program bude pomocí přepínačů fungovat v konzoli.

Je zřejmé, že žádný z programů uváděný v části 2.5 nesplňuje požadavky kladené na výslednou aplikaci – je tedy potřeba naprogramovat aplikaci vlastní. V následující části se vyjde ze stanovených stanovených cílů a navrhne model aplikace podle zadaných podmínek.

Programovací jazyk a rozšiřující knihovny

Po zvážení všech předchozích skutečností se ukázaly jako možné programovací jazyky C/C++, Java, Perl a Python. Z nich pak byl zvolen C/C++ a to jak z důvodů přenositelnosti, tak jednoduché možnosti rozšíření o potřebné externí knihovny. Pro jazyk C/C++ existuje možnost přilinkování dynamické knihovny pro komunikaci s LDAPem (parametr -lldap, pro C++ obdobně -lldapcpp). Pro ostatní programovací jazyky existuje též možnost rozšíření o LDAP client API, v tomto požadavku jsou možnosti rozšíření rovny.

Konzultacemi s vedoucím práce bylo vyloučeno pouhé zobrazení výsledků do konzole. Jako lepší se ukázalo použití knihovny ncurses, ta poskytuje možnost vykreslování do oddělených oken, scrolovatelné menu a další grafické funkce. Použitím knihovny ncurses se omezil výběr jazyků pouze na C/C++, ostatní jazyky podporu pro tuto knihovnu nemají nebo není úplná (nepodporují určité grafické prvky apod.). Výsledná aplikace bude naprogramována pomocí jazyka C/C++ s rozšířením o podporu přístupu k adresářové struktuře LDAP, knihovnu ncurses a podporou čtení XML souborů. Program bude po dohodě konfigurovatelný souborem ve formátu XML.

MVC

Model výsledné aplikace by měl být navržen s využitím architektury MVC (Model View Controler). MVC je velmi často využívan jako architektura softwaru. Rozděluje návrh aplikace na tři základní části, datový model aplikace (Model), uživatelské rozhraní (View) a řídící logiku (Controler).

- Datový model aplikace – v našem případě nebude datový model reprezentován serverem LDAP, ale objektem načtených dat LDAPu.
- Uživatelské rozhraní – textová nebo grafická část výsledné aplikace.
- Řídící logiku – zpracování událostí uživatele.

Postup zpracování je pak rozdělen na šest kroků, podobně jako uvádí zdroj [14]:

1. Uživatel provede akci (událost) v uživatelském rozhraní – například potvrzení dialogu nebo události klávesnice.
2. Řídící logika obdrží oznámení o této akci (událost je emitována) z objektu uživatelského rozhraní.
3. Řídící logika přistoupí k datovému modelu, v případě potřeby je aktualizován podle provedené uživatelské akce (definovaný slot pro událost).

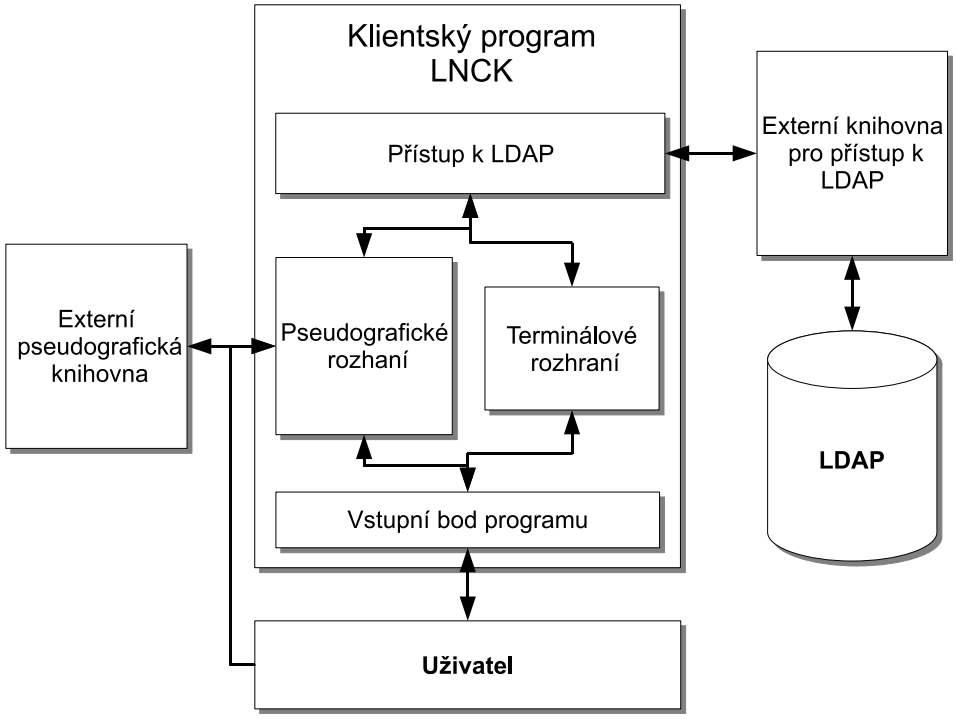
4. Datový model zpracuje přijímaná data a upraví svůj stav (např. přidání či editace atributu).

5. Komponenta uživatelského rozhraní použije zaktualizovaný datový model pro zobrazení dat uživateli. Uživatelské rozhraní získává data přímo z datového modelu, přičemž datový model nepotřebuje žádné informace o komponentě uživatelského rozhraní (je na modelu nezávislé, uživatelských rozhraní může být více).

6. Uživatelské rozhraní očekává další události uživatele, která celý cyklus zahájí znovu.

3.3 Návrh modelu

Nyní je možno sestavit návrh modelu výsledné aplikace. Uživatel bude přistupovat ke klientské aplikaci prostřednictvím vstupního bodu programu.



Obrázek 3.2: Model aplikace

Zde se budou parsovat parametry předané uživatelem a bude se rozhodovat o spuštění terminálového nebo pseudogafického rozhraní. V případě volby pseudografického rozhraní pak bude program pro zobrazení využívat externí knihovnu. Obě rozhraní pak v závislosti na uživateli přistoupí k adresářové struktuře LDAP pomocí části Přístup k LDAP a externí knihovny.

4 Implementace

V této kapitole se zaměříme na třetí bod zadání diplomové práce. Budou zde popisovány použité metody a jednotlivé kroky implementace, v závěrečné fázi pak uvedeny krátké úseky kódu a snímky obrazovky s běžící aplikací.

4.1 Simulace

Před začátkem vývoje aplikace bylo nutné simulovat podmínky cílového serveru na serveru lokálním. Nejprve je potřeba nainstalovat službu LDAP a importovat vzorová data. Pro účely testování byl využíván operační systém Fedora 9 firmy RedHat a server OpenLdap ve verzi 2.4.10-2 s nainstalovanou podporou serveru, klienta a vývojových prostředků, které využiji ve své aplikaci. Pro simulaci reálných dat v adresářové struktuře byla použita sada skriptů MigrationTools od firmy PADL Software Pty Ltd dostupný na adrese `http://www.padl.com/OSS/MigrationTools.html`. Dá se předpokládat, že se konfigurace lokálního serveru od serveru reálně nasazeného bude lišit jen minimálně.

Instalace na straně serveru

Jako zabezpečovací protokol je na školním serveru použit TLS (Transport Layer Security) využívající PKI infrastrukturu. Proto je třeba mít nainstalovánu podporu modulů autentizace (pam_ldap a nss_ldap) a vygenerované certifikáty. Využito je distribučního generátoru make-cert (místo běžného ručního vytváření nebo použití skriptu CA.sh). Skript vytvořil certifikační autoritu s kořenovým certifikátem podepsaným „sebou samým“ (v angl. selfsigned) a certifikát podepsaný touto certifikační autoritou. Konfigurace samotného serveru s ohledem na podporu TLS spočívala v úpravě konfiguračního souboru slapd.conf, konkrétně odkomentování řádků:

```
1 TLSCACertificateFile /etc/pki/tls/certs/ca-bundle.crt
2 TLSCertificateFile /etc/pki/tls/certs/slapd.pem
3 TLSCertificateKeyFile /etc/pki/tls/certs/slapd.pem
```

Další úpravy tohoto konfiguračního souboru spočívali např. v nastavení těchto hodnot:

```
database    ldbm  # typ databáze
suffix      "dc=nti.tul,dc=cz"   # kořen stromu
directory   /var/lib/openldap-data/  # adresář s databází
rootdn      "cn=root,dc=nti.tul,dc=cz"  # záznam superuživatele
rootpw      {MD5}JBfgUB0CjyVmz+pCrOVQBA==   # šifrované heslo
```

Při testování instalace byla použita databáze bdb (Berkley Database) místo předkonfigurované ldbm (LDAP DBM). Odezva práce na malém vzorku referenčních dat však byla totožná, a proto zde byla ponechána volba ldbm. Dále byly upraveny hodnoty vztahující se k požadovaném DIT, účtu administrátora, jeho heslu a cesty ke konfiguračním souborům a certifikátům. Šifrované heslo bylo získáno pomocí příkazu *slappasswd -h {MD5}* a výsledek použit v konfiguračním souboru.

Instalace na straně klienta

Dalším krokem bylo nastavit odpovídajícím způsobem konfigurační souboru *ldap.conf*. Tedy konfigurační soubor pro přístup klienta k LDAP.

```
base dc=nti.tul,dc=cz # bod přístupu
bind_timelimit 10 # maximální časový limit v sekundách
ssl start_tls  # nastavení protokolu
tls_cacert /etc/pki/tls/certs/ca-bundle.crt # umistění certifikátu
pam_password md5 # použita hashovací funkce
```

Simulace reálných dat

Pro otestování vyhledávání v LDAPu, bylo nutné naplnit strukturu referenčními daty. Tento účel splnil perlovský skript migrate_common.pl od výše zmiňované firmy PADL Software Pty Ltd, který vygeneroval LDIF soubor s informacemi o lokálních uživatelích lokálního testovacího serveru. Ty pak byly importovány do databáze příkazem: *ldapadd -D "cn=root,dc=nti.tul,dc=cz" -W -f /tmp/users.ldif*. Po tomto kroku již bylo možné testovat zabezpečené připojení k LDAP a běžné operace, jako jsou získávání, editaci a mazání záznamů pomocí konzolových nástrojů jako je ldapsearch, ldapadd atd.

4.2 Programování aplikace

Pro programovaní aplikace bylo použito editoru Emacs s rozšířením speedbar k ladění textové části GDB. Ladění pseudografické části programu probíhalo pomocí grafické nástavby GDB, programu Valkyrie.

Závislosti

Sestavení projektu je zajištěno souborem Makefile. Jako kompilátor byl nejprve zvolen gcc, později s nutností kompilace modulu konfigurace byl nutný přechod g++. K parsování XML byla využita odlehčená externí knihovna xmlParser autora Franka Vanden Berghena a to ve verzi V2.23 vydanou pod licencí BSD (což pro tento projekt znamená volné použití i úpravy zdrojového kódu). Další externí knihovnou použitou v tomto projektu je rozšiřující pseudografická knihovna. Vývoj vlastní knihovny s podobným rozsahem potřebných funkcí je nad rámec této diplomové práce. Takové knihovny již existují a vyvíjejí se již řadu let, proto bylo výhodnější využít některého volně dostupného projektu.

Po testování všech dostupných nadstaveb byla vybrána knihovna NDK++, Ncurses Developement Kit for C++ dostupná na adrese `http://sourceforge.net/projects/ndk-xx/`. Jedná se o zapouzdření knihovny ncurses pro jazyk C++ pod licencí GPL. Knihovna dále poskytuje základní dialogové okna, menu atd. Nevýhodou použití tohoto projektu je fakt, že poslední verze této knihovny byla vydána v roce 2003 (poslední patch vydán v roce 2005). V rámci implementace bylo potřeba upravit soubory Makefile pro kompilaci s aktuálním kompilátorem gcc a opravit chyby odhalené při překladu.

Bohužel však projekt obsahuje mnoho chyb a nedostatků, které byly odhaleny až při programování posledních částí pseudografického rozhraní. Některé chyby byly ošetřeny pomocí výjimek, jiné užitím alternativních grafických částí. Do knihovny byly dodatečně implementovány některé operace se seznamy (například mazání, návrat ukazatele prvku seznamu apod.).

Naopak použití tohoto projektu je spojeno s výhodami, které žádný jiný projekt rozšiřující ncurses neposkytuje (zvažovány byly projekty CDK a The Dialog). Zejména se jedná o systém ošetření událostí pomocí signálů a slotů v takové podobě, jak je známe například z grafického toolkitu Qt.

Samotné linkování výsledného projektu pak závisí na knihovnách:

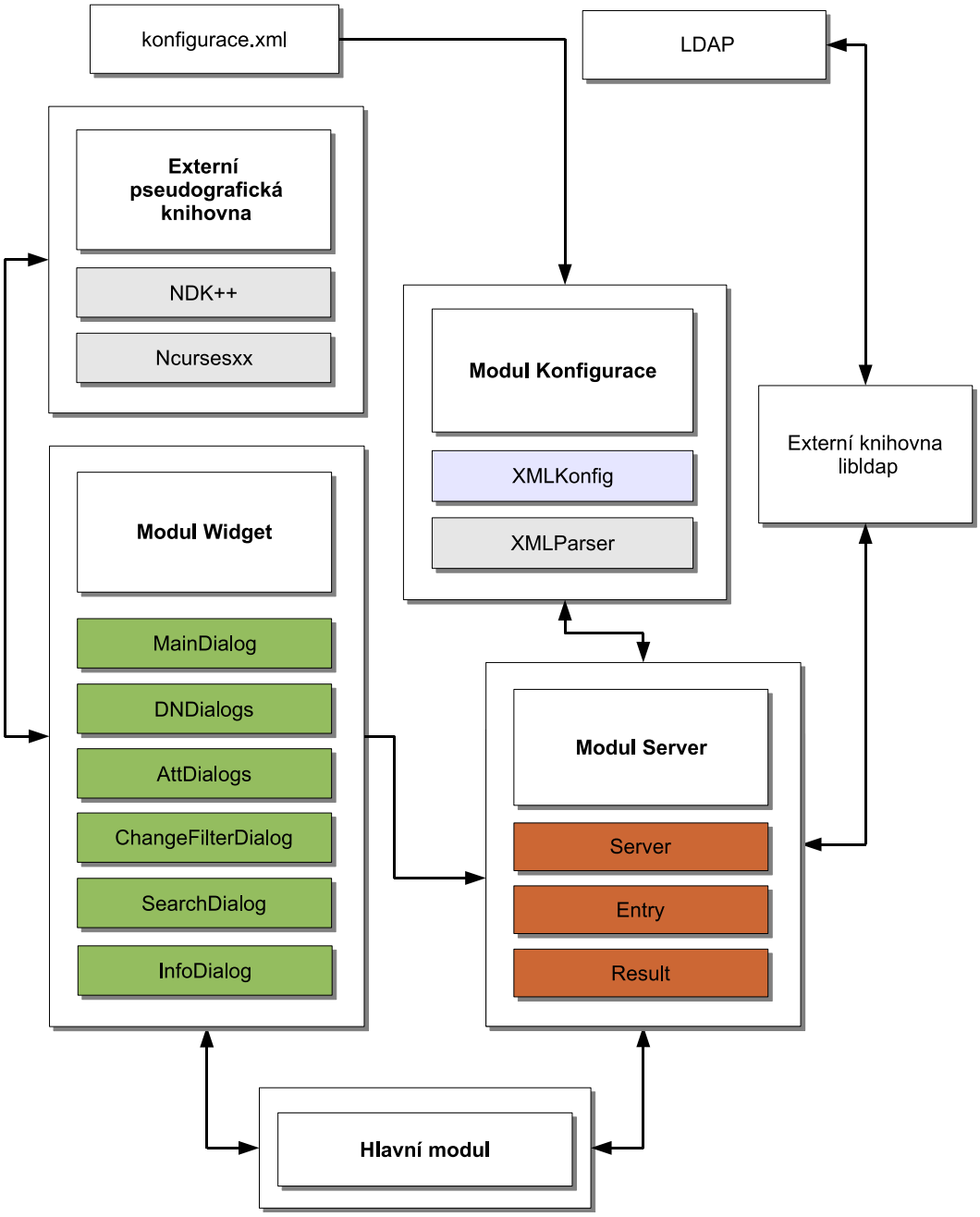
- lldap – přístup a komunikace s LDAP.
- lncursesw – podpora ncurses (inicializační funkce, přístup k barevným modelům a jiným konstantám). Dvojité v na konci názvu knihovny značí rozšířenou verzi knihovny pro multibajtové znaky[1], tedy podpora diakritiky.
- lmenu, lpanel, lform – podpora statických menu, panelů a možnosti přepínání mezi nimi a možnost použití formulářů.
- ndk++ – výše zmiňovaná pseudografická knihovna.

[1] angl. wide char, v kódu jsou pak tyto znaky typu wchar_t.

Zvláštností linkování je hodnota proměnné EXTRALDFLAGS ze souboru makefile. Je jím DLDAP_DEPRECATED, a to z důvodů zpětné kompatibility se staršími hlavičkami funkcí pro inicializaci komunikace s ldapem. Takto by měl být projekt přenositelný i na distribuce se staršími verzemi knihoven.

Výsledný kód

Vstupním bodem programu je hlavní modul. Ten reprezentuje třídu Main a její metodu *main*. Ta parsuje vstupní parametry programu pomocí standardní posix knihovny *getopt.h* a rozhoduje o dalším běhu programu.



Obrázek 4.1: Schéma modulů aplikace

Pokud je rozpoznána volba textového rozhraní, je ve funkci main alokován nový objekt server (modul server). Ten po své alokaci volá modul konfigurace. V tomto modulu je v části *XMLParser* načten konfigurační soubor *configuration.xml*, v části XMLKonfig jsou načtené hodnoty transformovány na cílové hodnoty zpracovávané LDAPem a vráceny zpět modulu Server. Ten pak pomocí externí knihovny pro přístup k LDAP provede připojení a vykonání funkce dle parametrů příkazového řádku.

V případě, že je v hlavním modulu rozpoznána volba pro zavedení pseudografického rozhraní, je alokován modul widget a podobně jako u textového rozhraní načteny informace z konfiguračního souboru. Uživateli je následně rozhraní zobrazeno a podle jeho vstupů jsou zobrazeny dialogové okna a volány metody modulu sever. Aplikace je ukončena po obdržení výsledků metod modulu server v textovém režimu nebo v pseudografickém po obdržení signálu quit.

Komunikace s LDAP

Komunikace se serverem LDAP probíhá ve dvou vrstvách. Textové i pseudografické jádro volá metody třídy Server. Tato třída obsahuje informace o spojení jako je adresa serveru, přístupové heslo nebo kořenový prvek, ke kterému se chceme připojit. Dále obsahuje všechny běžné metody pro práci s LDAP jako například *Server::attAdd* pro přidání atributu, *Server::loadSchemaResult* pro načtení vybraných částí schématu nebo *Server::init* pro nastavení spojení. Tyto metody tedy zapouzdřují nízkoúrovňové funkce v jazyce C. Vyšší vrstva přístupu k LDAP je implementována ve třídě WidgetMain. Metody této třídy přistupují ke třídě Server a pracují s výsledky jejích metod, ty jsou zobrazovány uživateli.

Dialogové okna

Pro potřebu této aplikace byla vytvořena sada dialogových oken. Každé z nich představuje dva hlavičkové soubory. První je definicí typů informací zobrazených v dialogu a druhá definuje vzhled a přístupové metody (get a set).

Zvláštním případem je okno hlavní. Klíčovým obsahem okna jsou dva panely. Zde bylo využito znalostí z předcházející kapitoly – vzhled aplikace je inspirován grafickým rozložením prvků v programu Gq popisovaným v části 2.5 a vzhledem známého souborového správce Midnight Commander. V levém panelu jsou zobrazovány záznamy získané prohledáváním podle kritérií z konfiguračního souboru a zadaného filtru. V pravém panelu se podle aktuálního záznamu zobrazují jeho atributy.

V horní části je menu pro vyvolání uživatelských dialogů, které pracují s informacemi o aktuálně zobrazeném záznamu a atributu. Ve spodní části se pak nachází tlačítka rychlé volby, obsahují nejpoužívanější volby dialogů včetně ukončovacího.

Možnosti příkazového řádku

Jedním ze základních požadavků na funkce aplikace bylo ovládání programu pomocí přepínačů. K očekávaným přepínačům jako jsou -s pro vyhledání v LDAP či přepínač -d pro smazání daného záznamu byl přidán i přepínač -b, který uživateli usnadňuje práci tím, že zkrátí zápis DN o bázi. Zde je přehled možných přepínačů:

Tabulka 4.1: Možné přepínače

Přepínač	Atributy	Význam
-s		Zobrazení výsledků vyhledávání
-a	DN <ATT=VALUE>	Přidání záznamu s atributy
-r	DN NEW_DN	Přejmenování záznamu
-d	DN	Odstranění záznamu
-l	DN	Výpis povolených atributů záznamu
-g		Start pseudografického rozhraní
-f	FILTER	Nastavení vstupního filtru
-b		Místo DN se bude uvádět RDN
-h		Nápověda

DN zde znamená rozlišovací jméno, <ATT=VALUE> je seznam dvojic jmen atributů následovanými symbolem rovno a hodnotou, vyskytovat se musí minimálně jednou. Pokud se program spustí bez parametrů, je ve stejném stavu jako s přepínačem g (zkratka graphic), tedy startuje do pseudografického rozhraní.

Konfigurace

Konfigurace programu je zajištěna pomocí xml konfiguračního souboru s pevně zadaným názvem *„configuration.xml“*. Celý modul konfigurace se skládá z externí knihovny pro parsování xml dokumentů a vytvořenou funkcí, která zavolá parser se správnými hodnotami a nastaví globální výstupní parametry. Soubor *configuration.xml* je uvozen hlavičkou s verzí XML standardu a kódováním. Jako název kořenového elementu byl zvolen „configuration“ a přiřazen jedinečný jmenný prostor odpovídající názvu projektu.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <configuration xmlns="www.tul.cz/nti/lnck">
```

Struktura tohoto XML je velmi jednoduchá, kořenový element obsahuje dětské elementy s názvem proměnné a hodnota je uložena v atributu value. Nyní bude uveden seznam přípustných elementů:

authMethod – Nastavení autentifikační metody pro přístup k LDAP serveru. Možné hodnoty:

- LDAP_SASL_SIMPLE
- LDAP_AUTH_SIMPLE
- LDAP_AUTH_SASL
- LDAP_AUTH_KRBV4

Příklad použití:

```
3 <authMethod value="LDAP_SASL_SIMPLE"/>
```

desiredVersion – je verze serveru, ke které se chceme připojit, dnes zpravidla *LDAP_VERSION3* další možné hodnoty jsou: *LDAP_VERSION2*, *LDAP_VERSION1*. Příklad:

```
4 <desiredVersion value="LDAP_VERSION3"/>
```

objectFilter – vyhledávací filtr, je zde možnost použití symbolu * a logických operátorů viz kapitola 2.

```
5 <objectFilter value="objectclass=*"/>
```

root – zobrazovaný kořen adresářové struktury, též použit jako báze. Příklad:

```
6 <root value="dc=nti.tul,dc=cz"/> >
```

ldapHost – adresa serveru LDAPu.

```
7 <ldapHost value="localhost"/>
```

rootDn – kořenový element stromu

```
8 <rootDn value="cn=root,dc=nti.tul,dc=cz"/>>
```

rootPw – heslo pro přístup k serveru.

```
9 <rootPw value="tajneHeslo"/>
```

Konfigurační soubor je ukončen uzavíracím tagem kořenového elementu *configuration*.

```
10 </configuration>
```

4.3 Vybrané části zdrojového kódu

Tato část kapitoly Implementace se bude zabývat klíčovými nebo zajímavými úseky kódu s krátkým popisem. Krácená místa budou označena výpustkou.

Načítání atributů – tato ukázka je část vyhledávací funkce *ldapSearch* umístěné ve zdrojovém kódu server.cpp. Před tímto ukázkovým kódem je volána funkce pro navázání spojení s LDAP (funkce init obsahující příkaz bind). Ve funkci ldapSearch jsou podle zadaných parametrů vyhledávány záznamy. Jako paměťová reprezentace záznamu slouží ve výsledné aplikaci třída *Entry* (zapouzdření seznamů s atributy a informacemi o záznamu), reprezentací všech výsledků je pak třída *Result* (obsahuje seznam instancí třídy Entry). V první části zdrojového kódu je vytvořen objekt tmp třídy *Entry*.

```
1  ...
2  Entry tmp = Entry(name); //tvorba nového záznamu
3  for(attr = ldap_first_attribute(ld, entry, &ber); attr != NULL;
4      attr = ldap_next_attribute(ld,entry, ber),countA++)
5  {
6    if((vals = ldap_get_values(ld, entry, attr)) != NULL)
7    {
8        for(int i = 0; vals[i] != NULL; i++)
9        {
10          if(vals[i][0]!=':') // atribut lze zobrazit
11             tmp.attrMap.insert(multi(attr,vals[i]));
12          else                // atribut je nutno kodovat base64
13             tmp.attrMap.insert(multi(attr,
14                                      base64_encode((unsigned char *)vals[i],
15                                      strlen(vals[i]))));
16       }
17       ldap_value_free(vals);
18   }
19   ldap_memfree(attr);
20 }
21 ...
```

Následně se v cyklu for prochází všechny atributy záznamu *entry* a ukládají se do proměnné *attr*. Knihovna libldap nám poskytuje dvojí přístup k hodnotě atributu, pomocí funkce ldap_get_values lze získat textovou podobu hodnoty atributu. Pro binární podobu se používá funkce ldap_get_values_len. Pro účely této práce je využit pouze textový přístup, binární hodnoty odlišené znakem dvojtečka na začátku hodnoty atributu jsou kódovány do Base64 a dále zpracovávány stejně jako hodnoty textové.

Konstruktor objektu Server – další ukázkou je konstruktor třídy Server, jediným parametrem je logická hodnota *t,* která označuje jestli je objekt konstruován v textovém nebo pseudografickém kontextu. Hlavní náplní konstruktoru je inicializace privátních třídních proměnných, například uváděné kořenový uzel DIT nebo vyhledávací filtr. Inicializaci provádí pomocí obektu XmlConfig, který parsuje hodnoty konfiguračního XML.

```
1  Server::Server(bool t):
2  tui(t)
3  {
4    XmlConfig * xml = new XmlConfig(strdup("configuration.xml"));
5
6    root = xml->getRoot();
7    objectFilter =  xml->getObjectFilter();
8    ...
```

Vysoká úroveň přidávání atributu – následující metoda slouží pro přidání nového rozlišovacího jména. Je umístěna ve zdrojovém kódu *widgetMain.cpp* a ilustruje práci s dialogovými okny. Na počátku jsou vytvořeny instance dialogů a pomocných struktur, získán Po získání potřebných dat metodou get je na objektu serveru volána metoda dnAdd. Po každé aktualizační operaci je (pokud to má smysl viz řádek 16) aktuální položka v menu přesunuta na nově vytvořenou. Dále je volána metoda stisku tlačítka pro správné překreslení údajů.

```
1  void widgetMain::dnAdd()
2  {
3    string dn = listbox_.current_item()->text(); //získání rozlišovacího jména
4    dnAddForm mb; //alokace formuláře
5    dnAddInfo d;  //alokace objektu pro zapuzdření dat
6    ...
7    mb.set(d);
8    if( mb() ) //konstruktor formuláře
9      {
10       dnAddInfo data = mb.get();   //načtení údajů z formuláře
11       multimap<string,string> optMap;
12       optMap.insert(pair<string,string>(data.oc_,data.value_));
13       s->dnAadd(data.dn_,optMap);    //volání nižší vrstvy
14       ...
15       int index = actualDnIndex(data.dn_);
16       if(index >= 0)
17       {
18               listbox_.set_current(index);
19           listbox_.at(index)->click(); //pokud
20       }
21     }
22 }
```

Práce se seznamem atributů – metoda fillAttMenu slouží pro naplnění seznamu atributů. Při ošetření událostí klávesnice a myši v menu záznamů (viz obrázek 4.6) je potřeba upravit informace v menu atributů podle aktuální položky. Objekt searched je instancí třídy Result, metoda getAttList vrací seznam artibutů daného záznamu. Získaný seznam je pak přidán do seznamu zobrazených položek.

```
1  void widgetMain::fillAttMenu(string dn)
2  {
3    list<string> vysledek = searched.getAttList(dn);
4    list<string>::iterator it;
5    for(it = vysledek.begin(); it != vysledek.end(); ++it)
6       attbox_.add((*it).c_str());
7    return;
8  }
```

4.4 Řešení chybových stavů

Ošetření chybových stavů bylo rozděleno na dvě úrovně:

- Chyby obdržené od OS – řešení pomocí výjimek (kritický kód je uzavírán do bloků try a catch). Chyby nemají grafické ztvárnění, ohlásí chybovou hlášku na standardní výstup a ukončí běh programu. Jedná se například o nevyhovění požadavku o alokaci paměti a podobně.
- Chyby programu a externích knihoven – řešení pomocí modulu server. Při volání této funkce je podle vnitřního stavu objektu *server* zobrazena chybová hláška na standardní výstup nebo pokud program pracuje v pseudografickém režimu, zobrazí se informační dialog s chybovou hláškou. V této funkci se rozlišují dva typy chyb:
  - Chyby návratových hodnot – aplikace reaguje pouze zobrazením informačního dialogu.
  - Kritické chyby – dojde k zobrazení informačního dialogu a po stisku tlačítka *OK* je běh aplikace ukončen.

4.5 Testování

Na testování aplikace se podílel vedoucí diplomové práce. Před testováním na reálném provozu byly využity výše zmíněné ladící nástroje 4.2 na odhalení skrytých chyb v programu (neoprávněné čtení paměti apod.). Testování na cílovém serveru vyžadovalo následující kroky:

- Instalace potřebných závislostí – program bylo nutné na cílovém stroji překompilovat, nutností byla instalace gcc a gcc-g++.
- Úprava konfiguračního souboru – přizpůsoben byl konfigurační soubor pro přístup na cílový server.
- Kompilace na 64bitovém serveru – do souboru Makefile byly přidány kompilační parametry *-m32,* tím se zajistil překlad aplikace pro třiceti-dvou bitovou platformu a následně mohl být slinkován se statickými knihovnami.

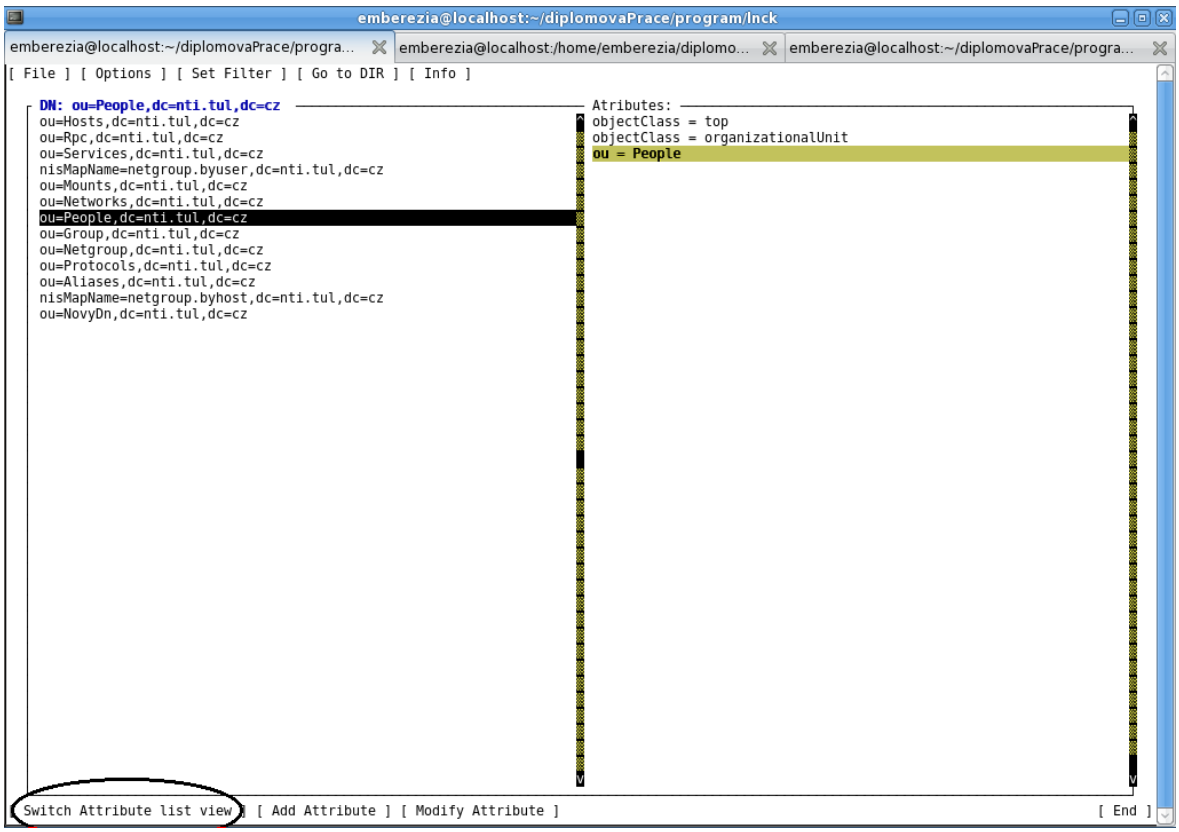
Při samotném testováni aplikace na reálných datech byly odhaleny chyby v zobrazení položek s diakritikou, nefunkční předávání filtru z příkazové řádky a mnoho dalších drobných chyb. Všechny nalezené chyby byly opraveny. Testování z hlediska bezpečnosti aplikace není třeba.

Tento program sice bude mít k dispozici heslo správce LDAP, ale fyzicky bude umístěn na zabezpečeném univerzitním serveru a klienti budou tento program spouštět přes vzdálené zabezpečené spojení SSH. Nehrozí tedy kompromitování hesla vlivem používání tohoto programu.

Aplikaci byla též experimentálně testována na časovou odezvu zpracování velkých objemů dat přijatých serverem LDAP. Načtení všech veřejných položek (vyhledávací filter *objectclass*=*) a jejich zpracování průměrně trvalo 1 minutu a 32 sekund. K testování byla využita stejná konfigurace PC jako pro lokální server. Počet položek byl cca 14 000. Čas reálného běhu programu nezávislého na běhu ostatních programů byl měřen programem *time* (součást projektu GNU). S výpisem na standardní výstup se program vykonával o 40 sekund déle, v pseudografickém režimu jen o 12 sekund. Prodleva při zpracování a následném zobrazení v pseudografickém režimu je tedy zanedbatelná.

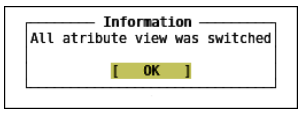
4.6 Snímky obrazovky

V této části jsou představeny snímky obrazovky výsledné aplikace při testování na referenčních datech. U každého snímku byly pro přehlednost invertovány barvy relevantních částí, tj. pozadí aplikace je ve skutečnosti černé. Na prvním obrázku je vidět hlavní část programu lnck – výsledky vyhledávání jsou zobrazeny do dvou panelů. Levý panel obsahuje přehled záznamů, které vyhovují aktuálně zadaným parametrům (nastavení v konfiguračním XML, případně vyhledávacímu filtru). V pravém panelu jsou zobrazeny atributy aktuálního záznamu. Aktuální záznam je označen inverzními barvami. Pokud je pracováno s nějakou pseudografickou částí (má tzv. fokus), je podbarvena modře tak jako aktuální položka zobrazeného seznamu atributů.



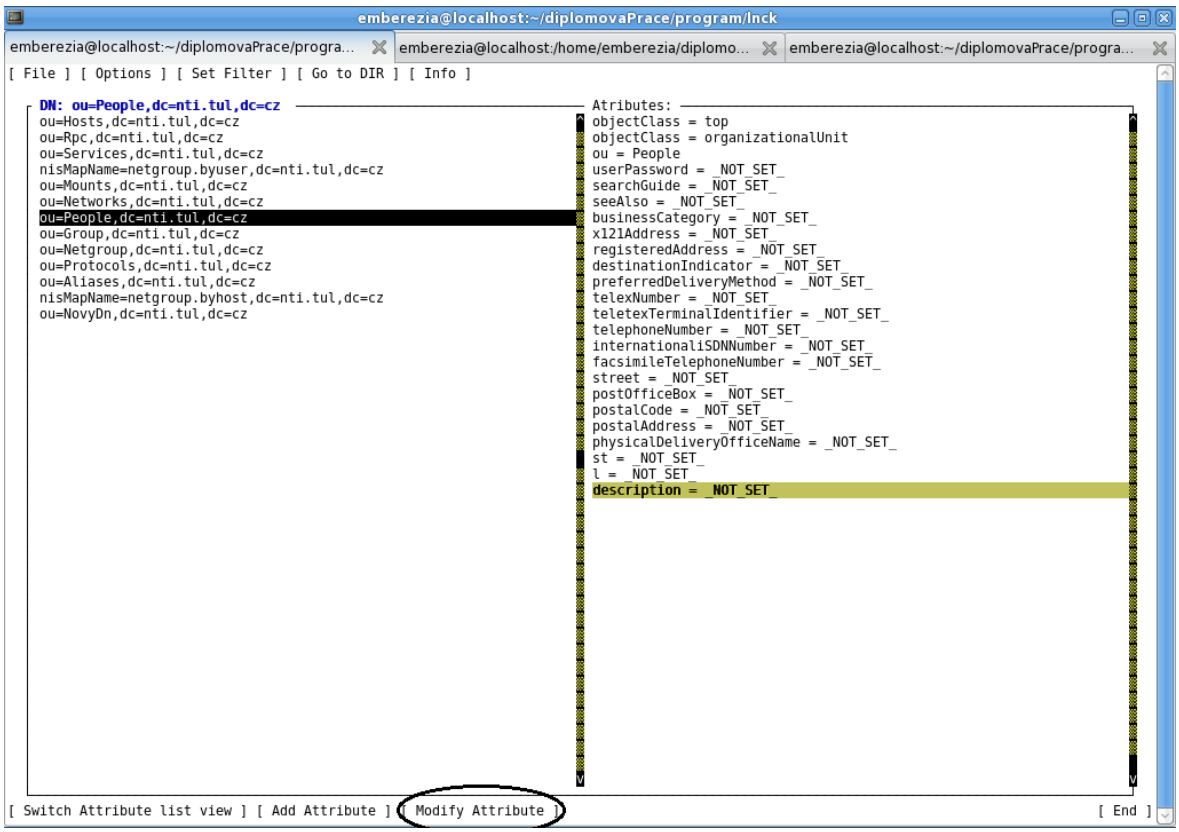
Obrázek 4.2: Zobrazení výsledků vyhledávání

Na obrázku 4.2 je černě zakroužkováno tlačítko *Switch atribute list view* sloužící pro přepnutí režimu zobrazení atributů. Po jeho stisku je za pomocí prohledání schématu přidáno k seznamu atributů seznam nevyplněných povolených atributů. Pro odlišení jsou zobrazovány s hodnotou *_NOT_SET_*, aby se zamezilo záměně mezi nevyplněným atributem a atributem s prázdnou hodnotou. Tento přepínač tedy slouží jako nápověda povolených atributů, po přepnutí se zobrazí následující informační dialog:



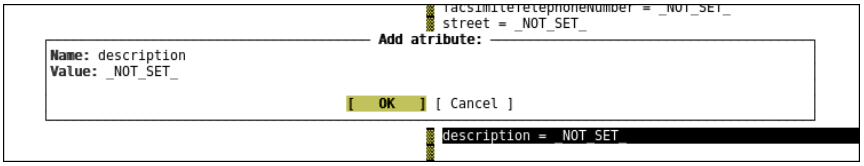
Obrázek 4.3: Oznámení o přepnutí stylu zobrazení atributů

Důvodem informování uživatele je ten, že je prodleva při vykonávání této operace závislá na počtu výsledných záznamů a odezvě LDAP serveru. Na obrázku 4.3 je vidět výsledek tohoto přepnutí. Zobrazeny jsou atributy s hodnotou, následuje seznam povolených atributů. Ty jsou získávaný tak, že se projdou všechny atributy *objectclass* daného záznamu a unikátně přidají do seznamu.



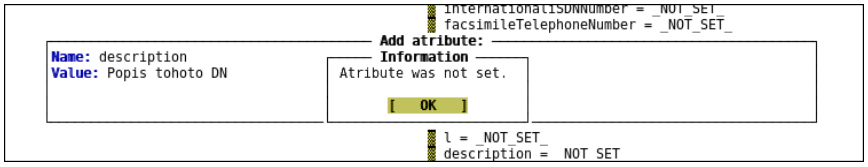
Obrázek 4.4: Zobrazení všech povolených atributů

Pokud administrátor potřebuje upravit některý z atributů, je zde možnost využití tlačítka *Modify atribut* (označeno černou elipsou) pro nastavení hodnoty konkrétní položky seznamu. Po úpravách atributů je vhodné přejít do běžného režimu zobrazení, tj. zobrazení pouze atributů s hodnotou, a to s ohledem na rychlost zobrazení i vytěžování LDAP serveru. Provádí se opětovným stiskem tlačítka *Switch atribute list view* na spodním panelu s tlačítky.



Obrázek 4.5: Dialog úpravy hodnoty atributu

Po stisku tlačítka *Modify atribut* je zobrazen dialog s vyplněným názvem atributu a jeho hodnotou. Vyplněná data jsou převzata z aktuální pozice v seznamu atributů.

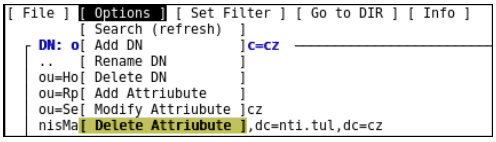


Obrázek 4.6: Stornování dialogu

Pokud stiskneme Enter případně klikneme myší na tlačítko OK, zavolá se vnitřní funkce pro modifikaci atributu. Následně je odeslán požadavek LDAPu na přidání atributu. Při úspěchu (LDAP vrací hodnotu *LDAP_SUCCESS*), se provede aktualizace zobrazených údajů a následné překreslení obrazovky. Při stisku tlačítka Cancel je dialog stornován s varováním.

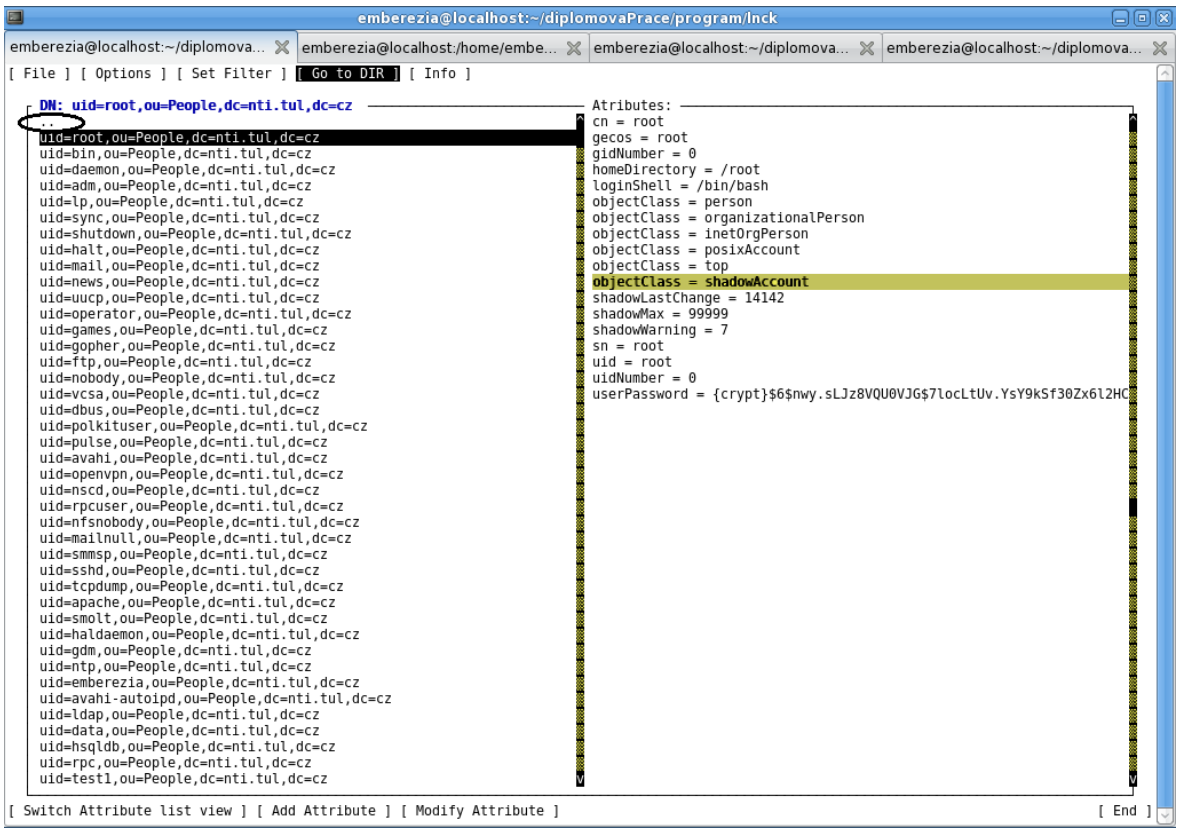
Podobně jako lze atribut modifikovat stiskem tlačítka *Modify atribut,* lze atribut přidat pomocí tlačítka Add atribute. Rozdíl mezi editací (modify) a přidáním (add) je na několika úrovních programu. V pseudografické části se v dialogu pro přidání atributu nevyplní aktuální atribut ze seznamu. Volána je pak metoda pro přidání atributu, která, podobně jako u editace, zapouzdřuje funkci ldap_add ovšem s parametrem pro přidání atributu. Tento rozdíl pak v aplikaci umožňuje přidávání multihodnotových atributů, které jsou následně zobrazeny jako oddělené atributy stejných názvů s různými hodnotami.

Aplikace však nabízí další možné volby a dialogová okna prostřednictvím horního panelu. Například tlačítko Options (Možnosti) zobrazí roletové menu s výčtem vyhledávacích a aktualizačních operací pro záznamy i atributy. V roletovém menu i celém panelu se lze pohybovat pomocí kurzorových kláves, klávesa *Esc* slouží k uzavření menu.



Obrázek 4.7: Menu

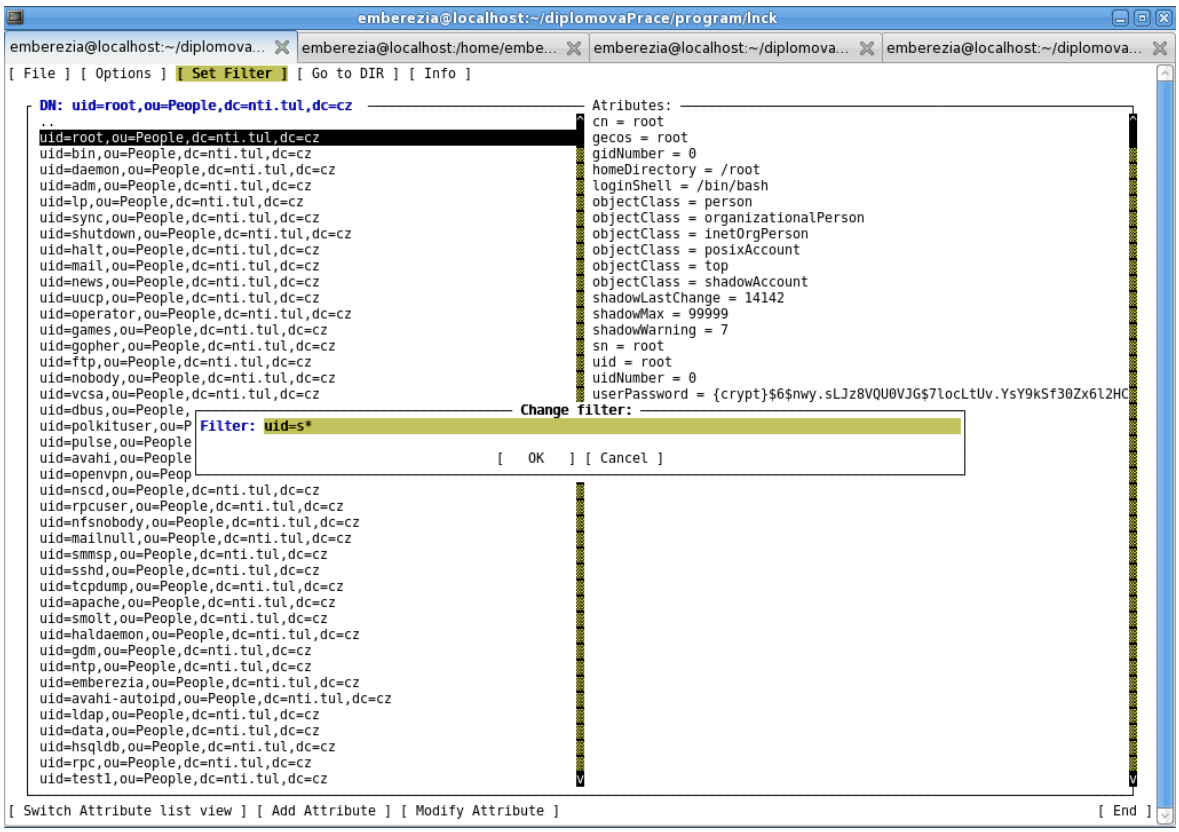
Funkce procházení adresáři je přiřazena tlačítku *Go to DIR*. Jak bylo již popsáno v části 2.4, prohledávání probíhá s nastavením *LDAP_SCOPE_ONELEVEL*. Nastavení tohoto parametru způsobí, že ze stromu výsledků je zobrazena jen jedna úroveň. Pro procházení těmito úrovněmi slouží právě tlačítko *Go to DIR* na daném záznamu. Po stisknutí tlačítka se testuje, zdali je záznam adresářem – tj. v DIT je uzlem a existují jeho listy, případně obsahuje další uzly. Pokud adresářem není, je uživatel informován zprávou, že daný záznam není adresář. V opačném případě je prohledán a jeho potomci jsou zobrazeny jako výsledky vyhledávání.



Obrázek 4.8: Procházení záznamem People

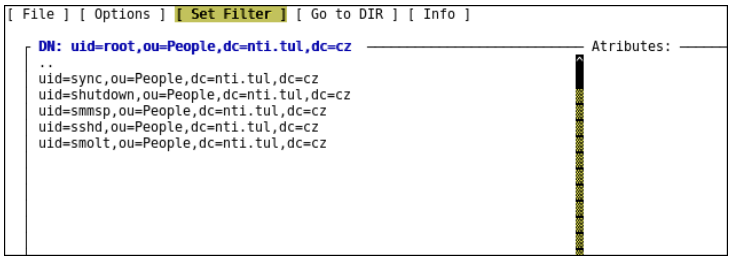
Na obrázku 4.8 je černě označena první položka v seznamu záznamů. Jedná se o uměle přidanou položku značící odkaz na rodičovský uzel. Zde byla snaha o náhled na adresářovou strukturu jako na souborový systém. Postup směrem výše je omezen na nejvyšší úrovni, tedy konfigurovanou hodnotou *base*.

Na hlavním panelu je dále možnost zvolit tlačítko File s volbou ukončení aplikace nebo tlačítko Info, které zobrazí informační dialog. Poslední možnost skýtá tlačítko *Filter*, které otevře dialog pro úpravu vyhledávacího filtru. Zobrazena je hodnota aktuálního vyhledávacího filtru s možností editace. Po startu aplikace nastaví modul server svou třídní proměnnou *filter* na hodnotu *objectFilter* z konfiguračního souboru. Iniciální nastavení je možno ovlivnit i parametrem *-f* v příkazovém řádku.



Obrázek 4.9: Snímek úpravy filtru

Obrázek 4.10 ilustruje změnu stávajícího filtru *objectclass*=* za *uid*=*s*. Po potvrzení je obrazovka překreslena s novým seznamem výsledků vyhledávání viz následující obrázek.



Obrázek 4.10: Výsledek změny filtru

5 Závěr

Závěrem bych rád zhodnotil dosažené výsledky, splnění zadaných cílů a využití výsledků v praxi. Jak bylo v úvodu práce předesláno, cílem bylo navrhnout a implementovat aplikaci pro přístup k adresářové struktuře LDAP.

V první kapitole této diplomové práce byl čtenář uveden do problematiky adresářové struktury LDAP. Dále byl podán přehled teoretických znalostí, norem a doporučení vztahujících se k dané problematice. Současně byly uvedeny dostupné funkční a programové prostředky. V následující kapitole došlo k návrhu modelu výsledné aplikace a výběru vhodného programovacího jazyka. V kapitole poslední byla popsána samotná implementace a testování výsledné aplikace.

Během vývoje aplikace se především v grafické části programu muselo čelit mnoha problémům, které jsou velmi jednoduše řešitelné například s pomocí grafických toolkitů Qt či GTK. S použitím pseudografické knihovny ncurses a jejích nadstaveb se však tyto problémy stávaly složitějšími a spíše se jednalo o zdlouhavé testování možných kombinací, které vedly k řešení.

Přínos této práce spatřuji v použití výsledné aplikace pro přehledné vyhledávání a editaci údajů adresářové struktury, přičemž vyhledávání výsledků je zobrazeno ve dvou panelech. Aplikace je díky použití knihovny ncurses použitelná i v případě vzdáleného přístupu k serveru přes SSH. Tím je zajištěno správné zobrazení i na jiných operačních systémech (například operační systém Windows v kombinaci s programem PuTTY). Vyvíjená aplikace nyní plně funguje na referenčních datech a splňuje všechny stanovené požadavky. Testována byla též v reálném provozu a po drobných úpravách je již připravena pro nasazení do praxe. Výsledný program se skládá z cca 7000 řádků vlastního kódu v jazyce C++ doplněný o externí knihovny. Samotná diplomová práce byla napsána v editoru LyX a sázena pomocí LaTeXu.

V budoucnu by se tato práce dala rozšířit na komplexní administrační nástroj pro LDAP s využitím stávajícího grafického základu. V programu by bylo možné kopírovat objekty, zálohovat do formátu LDIF nebo návrhovat a upravovat schémata. Další případné rozšíření této práce vidím ve vylepšení grafického rozhraní z hlediska interaktivity, zadávání konfigurace nebo tvorbu rozšiřujících dialogových oken.

Seznam tabulek

Seznam obrázků

Literatura

[1] SITERA, Jiří. *Technická zpráva číslo 2000-4 : Adresářové služby - úvod do problematiky*. CESNET [online]. 2000 [cit. 2009-05-02]. Dostupný z WWW: <http://www.cesnet.cz/doc/techzpravy/2000-4/>.

[2] DOSTÁLEK, Libor. *Velký průvodce protokoly TCP/IP : bezpečnost.* 2. aktualiz. vyd. [s.l.] : Computer Press, 2003. 592 s. ISBN 0-7226-849-X.

[3] TUTTLE, Steven, et al. *Understanding LDAP : Design and Implementation.* International Technical Support Organization. [s.l.] : Ibm.com/redbooks, 2004. 774 s. Dostupný z WWW: <http://www.redbooks.ibm.com/redbooks/pdfs/sg244986.pdf>. ISBN 073849786X.

[4] DOSTÁLEK, Libor, VOHNOUTOVÁ, Marta. *Velký průvodce infrastrukturou PKI : a technologií elektronického podpisu.* 1. vyd. Brno : Computer Press, 2006. 534 s. ISBN 8025108287.

[5] Mozilla LDAP C SDK Programmer's Guide Verze 16. 3. 2006 [cit. 2009-04-14]. Dostupný z WWW: <http://www.mozilla.org/directory/csdk-docs/>.

[6] JESUS, Castagnetto, et al. *PHP Programujeme profesionálně.* [s.l.] : Computer Press, 2001. 676 s. ISBN 8072263102.

[7] LDAP [online]. Wikimedia Foundation, Inc., 2009 , 2009 [cit. 2009-04-05]. Dostupný z WWW:<http://cs.wikipedia.org/wiki/LDAP>

[8] Base64 [online]. Wikimedia Foundation, Inc., 2009 , 2009 [cit. 2009-03-06]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Base64>.

[9] KLÍMA, Vlastimil. *Jak podepsat data : Moderní kryptografické metody.* Chip. 2000, č. 12, s. 62-65.

[10] The OpenSSL Project*: OpenSSL Documents [online].c1999-2009,* poslední úpravy 7.1.2009.[cit. 15.2.2009].Dostupný z WWW: <*http://www.openssl.org/docs*>.

[11] ITU-T Rec. X.680 (2002) | ISO/IEC 8824-1:2002. *Abstract Syntax Notation One (ASN.1) : Specification of Basic Notation* [online]. 2002. 2002, 2002 [cit. 2009-03-20]. Dostupný z WWW: <http://asn1.elibel.tm.fr/standards/>.

[12] SERMERSHEIM, J. Lightweight Directory Access Protocol (LDAP) : The Protocol. Network Working Group [online]. 2006 [cit. 2009-05-10]. Dostupný z WWW: <http://www.faqs.org/rfcs/rfc4511.html>.

[13] RFC [online]. Wikimedia Foundation, Inc., 2009 , 2009 [cit. 2009-04-07]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/RFC>.

[14] Model-view-controller [online].Wikimedia Foundation, Inc., 2009 , 2009 [cit. 2009-05-13]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Model-view-controller>.

6 Přílohy

[1] Výňatek z normy X.680

Číslo typu	Označení typu (hex.)	Typ
1	0x01	BOOLEAN
2	0x02	INTEGER
3	0x03	BIT STRING
4	0x04	OCTET STRING
5	0x05	NULL
6	0x06	OBJECT IDENTIFIER
7	0x07	OBJECT DESCRIPTOR
8	0x08	EXTERNAL
9	0x09	REAL
10	0x0A	ENUMERATED
11	0x0B	rezervováno
12	0x0C	rezervováno
13	0x0D	EMBEDDED
14	0x0E	rezervováno
15	0x0F	rezervováno
16	0x10	SEQUENCE a SEQUENCE OF
17	0x11	SET a SET OF
18	0x12	NUMERIC STRING
19	0x13	PRINTABLE STRING
20	0x14	TELETEXT STRING
21	0x15	VIDEO STRING
22	0x16	IA5 STRING
23	0x17	UCT TIME
24	0x18	GENERALIZED TIME
25	0x19	GRAPHICAL STRING
26	0x1A	VISIBLE STRING
27	0x1B	GENERAL STRING
28	0x1C	UNIVERSAL STRING
29	0x1D	rezervováno
30	0x1E	BASIC MULTILINGUAL PLANE STRING

Tabulka 6.2: Přehled X.680

[2] CD disk

Obsah přiloženého CD:

- Diplomová práce ve formátu PDF.
- Zdrojové soubory aplikace *lnck* se všemi externími knihovnami (postup instalace je v souboru INSTALL).